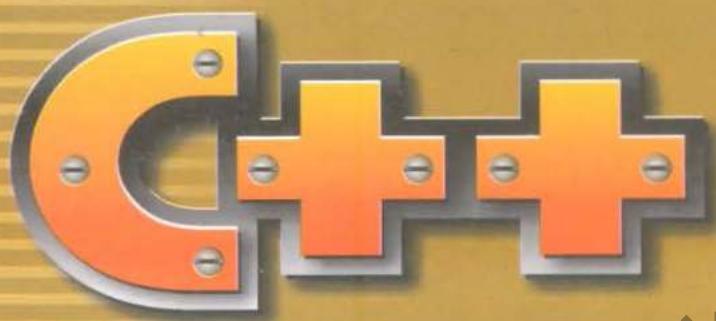
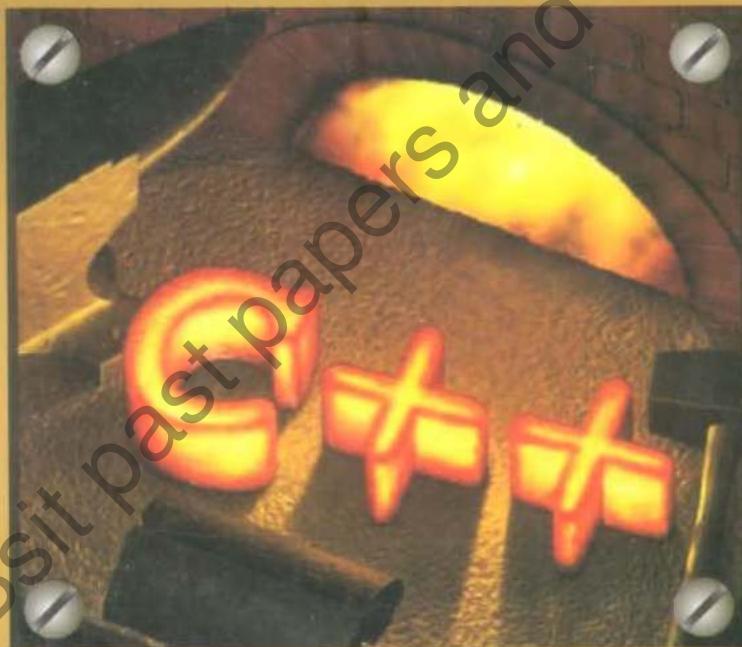


QUICK START SERIES



سیکھئ



محمد ذوالفقار نین چوہدری

JBD Press™

QUICK START SERIES

کچھ کتاب کے بارے میں

ہماری چند کمپیوٹر کتب

باب نمبر 1

اس میں C++ کی تاریخ اور جن عوامل کو مد نظر رکھتے ہوئے یہ لینکوں کی بنائی گئی ہے اس متعلق تمام معلومات شامل ہیں۔

باب نمبر 2

بیٹھا C++ کنٹرول شیٹ، بیٹھ لوپس اور فنکشنز شامل ہیں۔ یہ تینوں چیزوں ہر لینکوں کی وجہ پر ہوتی ہیں

باب نمبر 3

میں آپ اریزادو ٹنگ کے بارے میں تفصیل سے پڑھیں گے۔

باب نمبر 4

پاؤنٹر سے متعلق ہے۔ یہ C++ میں پہلی دفعہ شامل کیے گئے کیونکہ اصل C++ میں C سے derived قارم ہے اس لئے یہ اس میں بھی شامل کیے گئے ہیں۔

باب نمبر 5

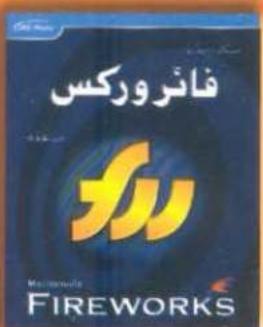
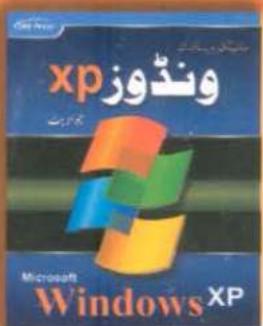
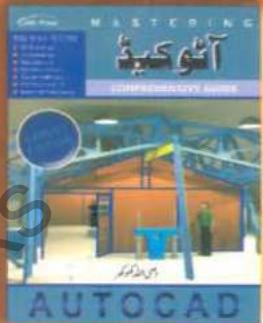
میں آپ او بجیکٹ اور بیٹنڈ پر گرامنگ میں کام کا شروع کریں گے۔ یعنی یہ سر کچرا اور کلاسز سے متعلق آپ کو معلومات فراہم کرے گا۔

باب نمبر 6

ایک اضافی فہرست کے بارے میں ہے جس میں آپ ہر آپریٹر سے اپنی مرضی کا آپریشن پر فام کر سکتے ہیں۔

باب نمبر 7

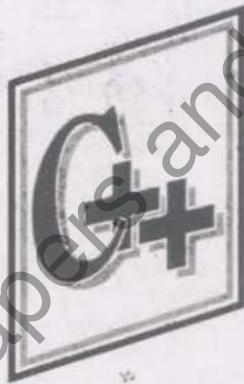
C++ کی جان ہے اس میں آپ پیور پر گرامنگ کے بارے میں پڑھیں گے۔



QUICK START SERIES

C++

سیکھئے



محمد ذوالقرنین چوہدری

جہانگیر بیک ڈپو

لاہور۔ راولپنڈی۔ کراچی

انتساب

چوبہری محمد اولیس

جن کی اعانت اور رہنمائی سے

میں نے یہ مقام حاصل کیا

Bsit past papers and books

حروف اول

ہم آپ کو پیور اوجیکٹ اور نیڈ پروگرامنگ لینکوں کج میں خوش آمدید کہتے ہیں۔ یہ کتاب خاص طور پر اوجیکٹ اور نیڈ پروگرامنگ کی بنیادی معلومات سیکھنے کے لئے ترتیب دی گئی ہے۔ گزشتہ چند سالوں سے ہارڈویر میں بہت تیزی سے ترقی ہو رہی ہے لیکن کچھ ناگزیر و جوہاں میں اس سافٹ ویئر میں خاطر خواہ ترقی نہیں ہو رہی۔ لوگوں کا سیکھنے کا رجحان ہارڈویر کی طرف زیادہ ہے کیونکہ سافٹ ویئر کی افادیت کا اندازہ نہیں اور نہ ان کو پروگرامنگ کے بارے میں بنیادی معلومات تفصیلًا فراہم کی جاتی ہیں۔ اس بات کو پیش نظر رکھتے ہوئے میں اوجیکٹ اور نیڈ لینکوں کے بارے میں ایک کتاب ترتیب دی ہے۔ اس میں آپ کو اوجیکٹ اور نیڈ پروگرامنگ کی افادیت اور ان کی بنیادی معلومات تفصیلًا فراہم کرنے کی کوشش کی گئی ہے۔ میں امید کرتا ہوں کہ یہ آپ کو پسند آئے گی۔

اس وقت دنیا میں سب سے زیادہ استعمال ہونے والی الحکومی اوجیکٹ اور نیڈ ہے لیکن بد قسمی سے پاکستان میں C++ میں اتنا زیادہ کام نہیں ہو رہا ہے۔ اس کی وجہ یہ ہے کہ نئی نسل کے اس لینکوں کے بارے میں کانسپیشن لیکس نہیں ہیں۔ اس کوڈ، ہن میں رکھتے ہوئے میں نے اس کتاب میں C++ کے بنیادی کانسپیشن سادہ لغزاں ان زبان میں سمجھانے کی کوشش کی ہے اور مجھے پورا یقین ہے کہ آپ تھوڑی سی محنت کے بعد بغیر کسی ٹاؤن کے اس کتاب کی مدد سے C++ لینکوں کی خود سے سیکھ لیں گے۔ اس کتاب میں کل سات ابواب ہیں اور ہر موضوع کو کم از کم ایک مثال کی مدد سے سمجھایا گیا ہے علاوہ ہر باب کے آخر پر مشتمل دی گئی ہے۔ جس میں روزمرہ زندگی سے متعلقہ سوالات اور بعد میں ان کے جوابات بھی تحریر مختصر ہیں۔ ایک اہم بات جس کا میں یہاں مذکورہ ضروری سمجھتا ہوں کہ آپ اس لینکوں کی جتنی زیادہ مشتمل یعنی پریکٹس کریں گے اس کے باعث میں آپ کے کانسپیشن اتنے ہی واضح ہوں گے اور مشہور کہاوت بھی ہے کہ "Practice makes a man perfect" اس کا امید کرتا ہوں کہ آپ کو یہ کتاب پسند آئے گی۔ آپ کے خیال میں اس میں اگر کمی و بیشی کی لمحائش ہے تو براہ مہربانی مجھے ضرور اطلاع دیجئے گا۔ میں آپ کا بے حد منون رہوں گا۔

چوبہری محمد ذوالقرنین

M.C.S, B.C.S, E-Commerce

zaki352@hotmail.com

گزارش

اس کتاب کو ترتیب دینے میں کئی لوگوں اور میرے اساتذہ کی معاونت مجھے درکار رہی ہے۔ ان میں سے بعض لوگوں نے پروگرامنگ سے متعلق اپنے تجربہ کی روشنی میں میری رخصائی کی۔ دوسروں نے اس ایڈیشن کا مطالعہ کیا اور اس کتاب کے مواد اور پروگرامنگ لینکوں کے نسبیت (Concepts) کو مورخہ لئے ہے بیان کرنے کے بارے میں اپنی مفید تجویز سے نوازا۔ میں اس سلسلہ میں خاص طور پر اپنے بڑے بھائی چوبہی محمد اولیس صاحب (المعلم نیاز صاحب) کا خاص ممنون ہوں۔

ان کے علاوہ میں ان احباب کا بھی مشکور ہوں جو گاہے بگاہے اپنی متعدد مشاورت سے مجھے نوازتے رہے۔ بالخصوص میں ان احباب کا بے حد ممنون ہوں۔

- میاں محمد اشرف صاحب (Director C.B.A)
- ظہیر احمد قریبی (Chairman)
- پروفیسر عبدالائق صاحب (M.I.S)
- اکرام الحق
- یاسر اورنگ زیب
- محمد الطاف حسین
- سید عظیم شاہ (M.C.S, OCP, M.C.S.D)
- محمد اکمل شہزاد (Computer Operator)
- پروفیسر سید کامران زیدی
- شتران احمد خان
- رضوان اکرم
- مس سیما کنوں
- چوبہری محمد شعیب

فہرست

13	• کتاب کے باریہ میں
15	پروگرامنگ کا تعارف C++ 1
16	- کمپیوٹر کیا
17	- لینوو بھر کی تاریخ
17	- C++ کی تاریخ
18	- C++ کی شینڈرڈ لا بھری
18	- C++ پروگرام کا تعارف
18	- C++ پروگرام لکھنا
21	- ویری استبلر
21	- کی ورڈز اور ایڈ سٹیشنیاٹر
21	- نیولائن کریکٹر
22	- ڈیٹا ناچس
23	- فٹٹا ناچ
24	- کریکٹر ڈیٹا ناچ
25	- فلوٹک پوائنٹ ڈیٹا ناچس
25	- حسابی Arithm آپریٹر
27	- آپریٹر کی فویت
27	- یوزری آپریٹر
28	- آرٹھمیٹیک آسانمنٹ آپریٹر
29	- ریلیشنل آپریٹر
30	- ناچ کنورڈن
32	- سرٹگ ان پٹ
33	- مشق
37	کنٹرول سٹرکچر اینڈ فنکشن 2
38	- کنٹرول سٹیٹ مینٹ
38	- if سٹیٹ مینٹ
39	- else-if سٹیٹ مینٹ

41	سٹیٹ مینٹ Nested-if ←
42	سٹیٹ مینٹ switch ←
45	- لا جیکل آپ پیئرز
46	- ویری ایبل سکوپ
48	- لوپس
48	- for
50 لوب Nested-for ←
52 لوب while ←
53	- لوپ do-while
54	- بریک سٹیٹ مینٹ
54	سٹیٹ مینٹ continue ←
55	سٹیٹ مینٹ goto -
56	- کانسٹ اینڈ او جیکلش
57	- شینڈرڈ میتھس فناشن
59	- یوزر ذیفائن فناشن
60	- ریٹن ناپ فناشن
62	- پیرا میٹر لست
68	- پیرا میٹر بائی ریفارن
70	- ریکریشن
72	- فناشن اور لوڈ نگ
73	- ذیفائن آر گومٹس
75	- مشق
85	ادیز اینڈ ستونگ 3
86	- اریز
89	- سرج Linear
90	- پائیزی سرج
91	- اریز کو ترتیب دینا

94	۔ ملٹی پل سکرپٹس ارے
98	۔ سڑنگز
98	۔ سڑنگ لاجیری
101	۔ سڑنگ جمع کرنا
102	۔ سڑنگ کاپی کرنا
106	۔ مشق

پوائیشنز

4

113	۔ تعارف
114	۔ ریفسنر
115	۔ پوائیشنر
116	۔ پوائیشنر آئینڈ اریز
118	۔ ریفسنر ریٹرن کرنا
119	۔ فٹشن پوائیشنر
121	۔ کریکٹ اور سڑنگ فٹشنر
123	۔ شیڈ آپ شٹر
125	۔ ڈیلیٹ آپ شٹر
126	۔ پوائیشنر زٹو پوائیشنر
126	۔ اریز
127	۔ مشق
129	

سٹرکچر اینڈ کلاسز

5

134	۔ سٹرکچر
135	۔ سٹرکچر اجزاء کو ایکسیس کرنا
135	۔ سٹرکچر پوائیشنر
138	۔ نیست نested سٹرکچر
140	۔ یوزر دیفارسند ڈیٹا نیچس
141	۔ کلاسز
144	۔ کلاس ڈیکلریشن
144	۔ پرائیویٹ اور پبلک کی ورثہ
146	۔ کنسٹرکٹر
149	۔ کنسٹرکٹر اور لوڈنگ
151	۔ اوبجیکٹ بطور آر گوٹس
153	

155.....	- نقش سے او بجیکٹ ریٹن کرنا
156.....	- ڈسٹرکٹ
159.....	- مشق
163.....	آپریٹر اور لوڈنگ 6
164.....	- فرینڈ نقش
165.....	- فرینڈ کاس
167.....	- کی ورڈ This
168.....	- ڈسٹیبل میر static
168.....	- نقش مبرز static
170.....	- آپریٹر اور لوڈنگ
170.....	- آپریٹر کی ورڈ
171.....	- اور لوڈنگ باائزی آپریٹر
173.....	- ملٹی پل اور لوڈنگ
175.....	- مشق
178.....	انہریتینس اینڈ پولی مار لیزر 7
179.....	- کپوزیشن
180.....	- انہریتیں
183.....	- انہریتیں کی اقسام
184.....	- اور رائیڈنگ مبر فناش
186.....	- کنسرکٹر اور ڈسٹرکٹ
188.....	- ملٹی پل کلاسز
189.....	- درچوک فناش
191.....	- کلاسز Abstract
193.....	- درچوک ڈسٹرکٹ
194.....	- انہریتیں کے لیواز
195.....	- ملٹی پل انہریتیں
198.....	- فائل پرو سینگ
202.....	- مشق

”کتاب کے بارے میں“

شروع اللہ کے نام سے جو بڑا امیران ہے۔

C++ لینکوچ کی اہمیت کو مد نظر رکھتے ہوئے میں نے کوشش کی ہے کہ اسی کتاب میں آپ کو C++ کی تمام بنیادی معلومات فراہم کی جائیں جو ایک پروگرام کے لئے ضروری ہوتی ہیں۔ دوسری کتاب میں اسی نسبت یہ کتاب بہت معیاری ہے۔ اس لئے کہ اس میں ہر بات ٹوڈی پواخت کی گئی ہے اور ہر عنوان یا C++ کا نہ ایک حقیقی زندگی پر مشتمل مثال کی مدد سے سمجھانا کی کوشش کی گئی ہے۔ اس کے علاوہ اس کتاب کا مودا بہت آسان رکھا گیا ہے تاکہ ہر خاص و عام کی سمجھیں آجائے۔ اس کتاب کے کل سات باب میں یہ سب کو ایک نظر دیکھتے ہیں کہ ان میں کیا ہے؟

باب نمبر 1:

اس میں C++ کی تاریخ اور جن عوامل کو مد نظر رکھتے ہوئے یہ لینکوچ بنال کے اس سے متعلقہ تمام معلومات شامل ہے۔ اس کے علاوہ اس میں C++ کا بنیادی انتزفیس بتایا گیا ہے کہ آپ کس طرح ایک آسان سا پروگرام بنانے کے لئے یہ کیا کروں۔

باب نمبر 2:

یہ باب بنیادی لحاظ سے کافی اہمیت کا حامل ہے کیونکہ اس میں C++ کنٹرول شیٹ سینٹ لوپس اور فلش حامل ہیں۔ یہ تیوں چیزیں ہر لینکوچ کی جان ہوتی ہیں اور ان کے بغیر آپ اپنی کوئی بھی اہم پر ابی حل نہیں کر سکتے۔ اس میں آپ کو بتایا گیا کہ کب اور یعنی آپ نے فیصلہ کرنا ہے اور پروگرام کی کوڈنگ کس طرح کرنی ہے۔

باب نمبر 3:

اس باب میں آپ اریز اور شرٹنگ کے بارے میں تفصیل سے پڑھیں گے۔ یعنی اریز میں سے کس طرح ویلیو حاصل کرتے ہیں۔ اریز کو ترتیب دینا پھر کوئی ویلیو اس کے درست آرڈر پر لکھنا اس کے علاوہ شرٹنگ ان پٹ کیسے لی جاتی ہے اور شرٹنگ کے ساتھ آپ کیسے آپریشن پر فارم کرتے ہیں۔ مثلاً دو شرٹنگز کو جمع کرنا یا کاپی کرنا توغیرہ۔

باب نمبر 4:

یہ باب پواخت سے متعلق ہے۔ پواخت Z C میں پہلی دفعہ شامل کئے گئے اور کیونکہ C++ اصل میں C کی ڈرائیوڈ فارم ہے اس لئے یہ اس میں بھی شامل ہیں۔ پواخت ز کافی مشکل ہیں لیکن ان کا فائدہ بہت ہے ان کی مدد سے آپ سرچ کا کام دوسرے پروگرام کی نسبت بہت آسانی سے کر سکتے ہیں۔ اس کے علاوہ فلشن آر گومینٹ Dynamic بنانے کے لئے آپ باب نمبر 4 میں پڑھیں گے۔

باب نمبر 5:

اس باب سے آپ اوجیکٹ اور نئیڈ پروگرامنگ میں کام شروع کر دیں گے۔ یعنی یہ سڑک پر اور کلاسز سے متعلق آپ کو معلومات فراہم کرے گا۔ اس میں پروگرامنگ اسی باب سے شروع ہوتی ہے۔ اس سے پہلے صرف بنیادی معلومات تھی جو پروگرامنگ کے لئے ضروری ہوتی ہے۔

باب نمبر 6:

یہ باب ایک اضافی فیچر کے بارے میں ہے جس میں آپ ہر آپ پر یہ سے اپنی مرضی کا آپریشن پر فام کر سکتے ہیں یعنی اس میں آپ آپ پر یہ اور اور لوڈ گکریں گے۔

باب نمبر 7:

یہ ہماری کتاب کا آخری باب ہے اور C++ کی جان ہے۔ اس میں آپ پور پروگرامنگ کے بارے میں پڑھیں گے۔ کہ پہلے سے بنی ہوئی چیزوں کو دوبارہ کس طرح استعمال کرتے ہیں یعنی اکیلیں ایک سافت ویر کو بار بار استعمال کرنے کے بارے میں بتایا گیا ہے اور اس کے علاوہ آپ اپنے ڈینا کو مستقل سور کرنے کا بھی طریقہ پڑھیں گے یہ باب انہائیشن (Inheritance)، پولی مارفزیم اور فائل پر وسیگ سے متعلق ہے۔ اس کے علاوہ ہم نے اس کتاب کے ہر باب کے آخر پر مشتمل کھصی ہے۔ جس میں حقیقی زندگی سے متعلق سوالات ہیں یعنی جو آپ روزمرہ زندگی میں ایک دوسرے سے پوچھتے ہیں اور بعد میں ان سوالات کے جوابتی درج ہیں۔

باب نمبر 1

C++ پروگرامنگ کا تعارف

ہم آپ کو C++ میں خوش آمدید کہتے ہیں۔ یہ C++ پروگرامنگ کا پہلا باب ہے اور اس باب میں آپ پڑھیں گے کہ C++ کیا ہے؟ اور یہ کیوں پڑھی جاتی ہے یعنی اسے پڑھنا کافی ہے؟ اس باب میں آپ C++ سے متعلق بنیادی معلومات حاصل کریں گے۔ C++ بہت مشکل لینگوچ ہے اور بب سے اہم یہ ہے کہ C++ میں بڑے حروف اور چھوٹے حروف کا خیال رکھنا بہت ضروری ہے یعنی یہ ایک Case sensitive لینگوچ ہے۔ اس باب میں آپ C++ کے مندرجہ ذیل بنیادی اصول پڑھیں گے۔

کریکٹرڈیٹیٹاپ		کپیوٹر کیا ہے؟	
فلونگ پوائٹ ڈیٹا ناپس		لینگوچر	
حابی آپ یئرز		C++ کی تاریخ	
آپ یئر کی فوکیت		C++ کی سینیڈرڈ لامبریری	
یئنڈی آپ یئر ز		C++ پروگرامز کا تعارف	
آرٹھمیک آجنتٹ آپ یئر ز		C++ پروگرام لکھنا	
ریلیشن آپ یئر ز		ویری اینبلر	
ٹاپ کونڑن		کی ورڈز اور ایڈیٹنگ فارمز	
ان پٹ		نیولائن کریکٹر	
سرنگ ان پٹ		ڈیٹا ناپس	
مشق		int	

کمپیوٹر کیا ہے؟

کمپیوٹر ایک الیٹر ایکٹ میشن ہے جو کو مختلف آپریشنز پر فارم کرتی ہے اس کے علاوہ یہ لا جیکل کام کرنے کے بھی اہل ہے اور یہ تمام کام ایک انسانی دماغ سے ملین یا بلین گناہ تیز کرتی ہے۔ جیسا کہ آج کل کا ایک کمپیوٹر کی سولین (ایڈیشنز) ایک سینڈ میں پر فارم کر سکتا ہے۔ کمپیوٹر ڈیٹا کو کچھ انٹرشن کے مطابق کنڑوں پر اس کرتا ہے اور یہ بدایات پر گرام کہلاتی ہیں۔ اس کے علاوہ کمپیوٹر کی فزیکل باڈی چھیپش سے مل کر بنتی ہے اور وہ یہ ہیں۔

ان پٹ یونٹ:

یہ کمپیوٹر کا معلومات مصل کرنے والا یونٹ ہے۔ اس میں کمپیوٹر مختلف ان پٹ ڈیٹا سس سے معلومات حاصل کرتا ہے اور اس معلومات کو اس طرح منتظم کرتا ہے کہ یہ معلومات بعد میں بھی پر اس کی جائیں۔ آج کل زیادہ معلومات کی بورڈ یا ماوس کی مدد سے کمپیوٹر میں سور کی جاتی ہیں اس کے علاوہ اب آپ بول کر بھی معلومات پیوٹیل میں سور کر سکتے ہیں۔

آؤٹ پٹ یونٹ:

یہ کیشن ان پٹ یونٹ کے بر عکس ہے۔ اس میں معلومات جو پر اس کی جا چکی ہوں وہ آؤٹ پٹ یونٹ پر ظاہر کی جاتی ہیں تاکہ نیوز راسے دیکھ سکے اور اگر وہ کمپیوٹر کے باہر بھی اسے نکالنا چاہتا ہے تو انکل سکے۔ مثلاً مائیکری پر شراس میں اہم کروار ادا کرتے ہیں۔

میموری یونٹ:

یہ کمپیوٹر کا عارضی دماغ ہوتا ہے۔ آپ ان پٹ یونٹ کی مدد سے جو معلومات کمپیوٹر کو فراہم کرتے ہیں وہ یہ معلومات میوری میں محفوظ کر لیتا ہے تاکہ بعد میں جب اس کو پر اس کرنے کی ضرورت پیش آئے تو آپ کو دوبارہ اس معلومات فراہم نہ کرنی پڑیں بلکہ کمپیوٹر سے ہی حاصل کر لیں اور اس کے علاوہ جب تک آپ پر اس کی ہوئی انفارمیشن کو آؤٹ پٹ پٹ یونٹ پر ٹیکس لے جائے تو اسکی اسکی یونٹ میں محفوظ رہتی ہے۔

آرچیٹیکٹ ایڈیٹ لاجک یونٹ (ALU):

یہ وہ یونٹ ہے جس میں آپ کا کمپیوٹر ڈیٹا کو پر اس کرتا ہے۔ اس میں کیلکولیشن مثلاً ایڈیشن، ٹائمینگ، غیرہ کی جاتی ہے۔ اس میں ایک سٹم ہوتا ہے جو کہ خود سے فیصلہ کر سکتا ہے۔ مثلاً دونسروں کا موازنہ کرنا ہے کہ کیا وہ برابر ہیں یا نہیں وغیرہ۔

سینٹرل پر اسینٹگ یونٹ (CPU):

یہ تمام کمپیوٹر کا ہیڈ ہے اور یہ دوسرے سیکشن کی مگرائی کرتا ہے کہ انہوں نے کون کون سے آپریشنز پر فارم کئے ہیں اس سے ان پٹ یونٹ کو بتاتا ہے کہ کب میموری یونٹ سے معلومات حاصل کرنے ہے اور کب ALU کو اسے پر اس کرنا ہے۔ غرض کہ یہ تمام دوسرے یونٹس کو آرڈر دیتا ہے کہ فلاں کام کرو۔

سینکڑری سور تھج یونٹ:

یہ میموری یونٹ سے کہیں بڑا ہوتا ہے اور یہ کمپیوٹر کا مستقل دماغ ہے جس میں آپ تمام ڈیٹا خواہ وہ پر اس ہو چکا ہے یا نہیں، محفوظ کر سکتے ہیں۔ مثلاً بارڈ ڈسک پر ڈیٹا سور کرنا وغیرہ۔

آپ نے اوپر کمپیوٹر کے بارے میں پڑھا ہے کہ وہ کس طرح کام کرتا ہے۔ آپ نے اس کے علاوہ Personal computers کا لفظ بھی سنا ہوا ہے یہ 1971ء میں اپنے کمپیوٹر والوں نے متعارف کر دیا تھا۔ اس سے کمپیوٹر بہت ستا ہو گیا اور اب اسے لوگ آسانی سے اپنے ذاتی کام کے لئے خرید سکتے ہیں۔ پھر 1981ء میں IBM نے پر سٹل کمپیوٹر متعارف کر دیا۔

لینکو بھر:

ہماری یہ کتاب C++ سے متعلق ہے اور میرا خیال ہے کہ آپ کو اس سے پہلے عام لینکو بھر کے بارے میں بھی معلومات حاصل ہونی چاہئے کہ C++ کوان پروفیشنل کیوں دی جاتی ہے۔ پروگرامر ز مختلف لینکو بھر میں کمپیوٹر کے لئے پروگرام لکھتے ہیں۔ ان میں سے کچھ لینکو بھر میں لکھا ہوا کوڈ یا پروگرام کمپیوٹر کو بھئے کے لئے اسے کسی ٹرانسیلیٹر کی ضرورت ہوتی ہے۔ بالکل ایسے ہی جیسے آپ کی ملک میں جائیں جہاں کی زبان آپ کو نہ آتی ہو تو آپ کو ایک مترجم کی ضرورت ہو گی جو آپ کوان لوگوں کی اور ان لوگوں کو آپ کی زبان سمجھائے گا۔ آجکل بہت زیادہ کمپیوٹر لینکو بھر استعمال ہو رہی ہیں۔ ان کو تین حصوں میں تقسیم کیا گیا ہے۔

Machine Language

Assembly Language

High-level Language

میں لینکو بھر نام سے ظاہر ہے کہ یہ کمپیوٹر حصوںی زبان ہی ہو گی اور ایسا ہے بھی کیونکہ کمپیوٹر صرف اس لینکو بھر کوڈ ازیکٹ بغیر کسی ٹرانسیلیٹر کے سمجھتا ہے اس میں کوڈ صرف '0's اور '1's میں لکھا جاتا ہے۔ لینکو بھر میں پر انحصار کرنی ہیں کیونکہ یہ میں کے ہارڈ ویئر کے مطابق ڈیفارس کی جاتی ہیں۔ لینکو بھر انسانی دماغ کے لئے سمجھتا ہے مثکل ہیں۔

جیسے جیسے وقت گزرتا گیا کمپیوٹر میں بھی ترقی ہوئی گئی اور پر زیادہ تر پروگرامر نے یہ محسوس کیا کہ میں لینکو بھر بہت سست اور مشکل ہے۔ اس لئے پروگرامر نے سڑک نمبرز کی بجائے انگلش کے حروف استعمال کرنا شروع کر دیا۔ جن کی بنیاد پر پھر ایکسلی لینکو بھر بنائی گئی۔ پھر بعد میں اس لینکو بھر کو کمپیوٹر کے لئے آسان اور تیز بنانے کے لئے ٹرانسیلیٹر بھی بنایا گیا جس کو اسیبلر (Assembler) کہتے ہیں۔ اس میں لکھا گیا کوڈ انسان کو آسانی سے سمجھا سکتا تھا لیکن اس کی رفتار بہت ستھی اس لئے کہ یہ پہلے میں لینکو بھر میں کوثر بیان کیا جاتا ہے اور پھر یہ مطلوبہ کام کرتا تھا۔

اس کے بعد پھر High Level لینکو بھر بنائی گئیں۔ ان میں ایک کام آپ صرف ایک سٹریکٹ میں پر فارم کر سکتے ہیں۔ اس لئے ان کی دوسری لینکو بھر کی نسبت سپیڈ بہت زیادہ ہے اور ٹرانسیلیٹر پروگرام جو اس لینکو بھر کو میں لینکو بھر میں کوثر کرتا ہے وہ کمپائلر کہلاتا ہے۔ C++، لینکو بھر بھی ایک ہائی لیول لینکو بھر ہے۔ جو کہ تقریباً اس سے زیادہ پاور فل اور استعمال ہونے والی لینکو بھر ہے۔ کمپائلر کے بعد پھر پروگرامر نے اسٹر پیٹر پروگرام بنایا جو کہ ہائی لیول لینکو بھر کوڈ کوڈ ازیکٹ ایگزیکیوٹ کرتا ہے۔ اس طرح اسے پہلے کمپائلر کی مدد سے میں لینکو بھر میں کوثر کرنے کی ضرورت نہیں ہو گئی اور اس کی سپیڈ مزید تیز ہو گئی۔

C++ کی تاریخ:

1967ء میں مارٹن رچڈ نے دو لینکو بھر BCPL اور B.BCPL کھیس جو کہ آپ یہنگ سٹم اور کمپائلر بنانے کے لئے آئندی میں تصور کی جاتی تھیں۔ 1970ء میں کیون تھا میں نے اپنی لینکو بھر B میں کئی اہم فچرز کا شاہد کیا کیونکہ وہ ایک آپ یہنگ سٹم بنانا چاہتا تھا۔ اسی سال کے اختتام پر اس نے بیل لیبارٹری میں یونیکس (UNIX) آپ یہنگ سٹم کا پہلا درودن تیار کیا۔

پھر اس کے بعد ڈنیس رچی (Dennis Ritchi) (Dennis Ritchi) نے بیل لیبارٹری میں B لینکو بھر کی مدد سے ایک بنی لینکو بھر متعارف کر دیا جس کا نام C رکھا گیا۔ اور یہ پہلی دفعہ 1972ء میں مارکیٹ میں متعارف کر دیا گی۔ C میں B اور BCPL لینکو بھر کے کئی اہم فچرز شامل ہیں۔ C سے یونیکس آپ یہنگ سٹم بنایا گیا جس کی وجہ سے یہ بہت مشہور ہو گئی آجکل زیادہ تر آپ یہنگ سٹم C میں ہی لکھے جا رہے ہیں۔

آپ نے سا ہو گا کہ انسان نے چاند کو تینی کر لیا ہے لیکن پھر بھی اسے سکون نہیں ہے وہ زیادہ سے زیادہ کامیابیاں حاصل کرنا چاہتا ہے۔ اسی طرح پروگرامر نے بھی محسوس کیا کہ C میں وہ تمام کام جیسیں کر سکتے جن کی انہیں ضرورت ہے۔ اس کے پیش نظر 1980ء میں بیل کی لیبارٹری میں بھارن

سٹر و سٹرپ (Bjarne Stourstrup) نے C++ لینگوچ کی ایڈوانس فارم ہے۔ C++ میں C کی نسبت کئی اہم فچر شامل کئے گئے اور اس کے مشہور ہونے کی سب سے سے بڑی وجہ او بجیکٹ اور بینڈ پروگرامنگ ہے۔ او بیکش ایسے سافٹ ویز کپوئیٹ ہوتے ہیں جنہیں آپ بار بار استعمال کر سکتے ہیں۔ او بجیکٹ اور بینڈ پروگرام سمجھنے اور دوبارہ ایڈٹ کرنے میں آسان ہوتے ہیں اور یہ بالکل درست کام کرتے ہیں۔

C++ سینڈرڈ لائبریری:

آپ نے اوپر C++ کی تاریخ کے بارے میں پڑھا کہ یہ کب اور کس مقصد کے لئے بنائی گئی۔ ہم نے آپ کو بتایا کہ C++ اصل میں C کی ایڈوانس فارم ہے جو اس میں C کے فچر کے علاوہ اضافی فچر بھی شامل کئے ہیں۔ C++ پروگرام کا سر اور فناش پر مشتمل ہوتے ہیں۔ C++ کی سینڈرڈ لائبریری پہلے سے جوئے فناش اور کا سر کی ایک بڑی مقدار فراہم کرتی ہے۔ اس طرح آپ کو C++ میں یہ سیکھنا بہت ضروری ہے کہ اس کی سینڈرڈ لائبریری کس طرح انہماں کر سکتے ہیں۔

C++ پروگرامز کا تعارف:

C++ کے پروگرام مختلف چھ مراحل میں حصہ کر آپ کو آؤٹ پٹ ڈپلے کرتے ہیں۔ یہ مختلف چھ مراحل مندرجہ ذیل ہیں۔

ایڈٹ، پری پر اس، کپائل، لانک، ادھ اور ایگزیکیوٹ

پہلا مرحلہ ایک فائل کو ایڈٹ کرنے کا ہے۔ یہ کام آپ سے ایڈٹ پروگرام میں کرتے ہیں۔ یعنی اپنا پروگرام ناٹپ کرتے ہیں اور ضروری غلطیاں ختم کرتے ہیں۔ اس کے بعد پروگرام مزید بعد میں استعمال کرنے کے لئے ہر ڈسک پر کہیں بھی شور کر دیا جاتا ہے۔ آپ کے C++ پروگرام کی ایکس ٹینشن.cpp ہونا ضروری ہے۔

اس کے بعد پروگرام کپائل کیا جاتا ہے۔ اس میں C++ کپائلر C++ و مشن لینگوچ کوڈ میں ٹرانسلیٹ کرتا ہے۔ اس کے بعد پروگرام کو سینڈرڈ لائبریری یا اگر آپ نے کہیں اور سے کا اس یا فناش استعمال کیا ہے تو اس کے ساتھ لہجہ کرنے کا مرحلہ آتا ہے۔ اس کے بعد لوڈنگ پوائنٹ ہے۔ کوئی بھی پروگرام ایگزیکیوٹ کے جانے سے پہلے یہیں اسکی لوڈ کیا جاتا ہے اور یہ کام لوڈ رکرتا ہے اور آخر میں آپ کا پروگرام ایگزیکیوٹ ہوتا ہے۔ اگر اس میں کوئی غلطی نہیں ہوگی تو یہ ایگزیکیوٹ ہو گا ورنہ لائز نہیں ایسا جب ایردے گا تو وہ غلطی درست کریں اور بعد میں اپنا پروگرام ایگزیکیوٹ کریں۔

C++ ایک مشکل لینگوچ ہے اس لئے اس میں آپ کو زیادہ سے زیادہ مہارت حاصل کرنا ہوگی تب آپ ایک اچھا پروگرام لکھ سکیں گے۔ اس کتاب میں ہم نے کوشش کی ہے کہ آپ C++ کی بنیادی چیزیں زیادہ سے زیادہ سیکھ سکیں۔

C++ پروگرام لکھنا:

Lینگوچ کپیوٹر پروگرام کا ایک واضح اور منظم ذیزان مہیا کرتی ہے یعنی اس میں لکھے ہوئے پروگرام کا خالک بہت منظم ہوتا ہے۔ جیسا کہ ہم نے پہلے بتایا ہے کہ C++ کا کپائلر سورس فائل کو پر اس (کپائل) کرنے کے بعد ایگزیکیوٹبل فائل میں تبدیل کر دیتا ہے جسے آپ اپنے کپیوٹر میں دوسرا پروگرامز کی طرح چلا سکتے ہیں۔ اس میں شامل سورس فائل اصل میں نیکست فائلز ہوتی ہیں۔ ان کی ایکس ٹینشن.exe (Executable) فائلز کی ایکس ٹینشن.exe ہوتی ہے۔

ہم نے یہچاں باب میں C++ پروگرامنگ کی کچھ بنیادی مثالیں تحریر کی ہیں جن میں C++ کے اہم بنیادی فچر کا استعمال کیا گیا ہے۔ ایک پروگرام کچھ ہدایات (انٹر کشن) کا جمود ہوتا ہے۔ جسے ایگزیکیوٹ کیا جا سکتا ہے۔ آئیے C++ کا ایک ساداہ پروگرام دیکھتے ہیں۔

مثال نمبر 1.1 میں

```
# include <iostream.h>
void main( )
{
    cout << "Welcome to C++ Program:" ;
}
```

یہ C++ کا ایک بسیار سادہ پروگرام ہے۔ اس پروگرام کی پہلی لائن یہ ہے۔

```
# include <iostream.h>
```

کسی بھی فرم کی آؤٹ پٹ کھوائے کے لئے اس لائن کا لکھنا ضروری ہے۔ اس میں # include کو پری پریس ڈائریکٹو (Preprocessor Directive) کہتے ہیں۔ اس میں # سب سے پہلے پڑھا جاتا ہے۔ یہ ڈائریکٹو ایک سٹائل فائل iostream.h کو ریفر کرتا ہے۔ جس میں cout یعنی ان پٹ آؤٹ پٹ کے بارے میں معلومات محفوظ ہوتی ہے۔ (h)۔ اس پٹ کی نشانہ ہی کرو رہا ہے کہ یہ ایک ہیڈر فائل ہے جس میں پہلے سے فناشز یا ورڈز کے بارے میں معلومات محفوظ ہیں۔ اور ایک اہم بات کہ <اور> بریکس اسی فائل کے نام کا حصہ نہیں ہیں بلکہ اس کے بعد دوسرا لائن void main() ہے۔ اس کو ہیڈر فناشن (main) کہتے ہیں اور یہ ہر C++ پروگرام کے لئے لکھنا ضروری ہے۔ یہ C++ کپیلر کو یہ واضح کرتی ہے کہ پروگرام کہاں سے شروع ہوتا ہے۔ اس میں یہ () بریکس لکھنا بھی ضروری ہیں یہ فناشن کے لئے لکھی جاتی ہیں۔ اس کے بعد یہ () بریکٹ ہے جو سب سے آخر میں بند (بھی ہو رہی ہے۔ کسی بھی فناشن کے لئے ان () بریکس کا لکھنا ضروری ہوتا ہے۔ یہ () main کی باڑی کی نشانہ ہی کرتی ہیں۔ () پروگرام کا یہ حصہ ہے اس کے بعد یہ لائن ہے۔

```
cout << "Welcome to C++ Program:" ;
```

یہ لائن cout اور بیکٹ کو یہ "Welcome to C++ Program" سٹیٹ میٹ بھجنے کے لئے استعمال ہو رہی ہے۔ آپ اس کی جگہ پر کچھ بھی پرنٹ کرو سکتے ہیں۔ اس میں cout میئنڈرڈ آؤٹ پٹ شریم ہے جو کوئی بھی لائن کپیلر کی پرڈسپلے کروانے کے لئے استعمال ہوتی ہے اور کم خفف ہے۔ اور آپ دیکھ رہے ہوں گے کہ ہم نے اس میں مطلوب الفاظ " میں تحریر کئے ہیں۔ اس سے واضح ہوا کہ اپ جو بھی الفاظ ڈسپلے کروانا چاہتے ہیں انہیں " میں تحریر کریں وہ بالکل اسی فارمیٹ میں سکرین پر ڈسپلے ہو جائیں گے جس طرح آپ نے لکھے ہیں۔ اور اس کے آخر میں یہی کالن (:) سے یہ ہر سیٹ میٹ کے بعد لکھنا ضروری ہے اور جب تک آپ یہیں لکھیں گے کیا لاء جو لائن کو ایک ہی سٹیٹ میٹ تصور کرے گا یعنی یہ ایک سٹیٹ میٹ کے اختتام کی نشانہ ہی کرتا ہے۔ cout کے بعد <> علامت کو آؤٹ پٹ آپریٹر یا انترتیشن (Insertion) آپ پر ایثر کرتے ہیں۔ یہ آپ کی تحریر cout کو منتقل کرتا ہے۔ آپ نے C++ میں پروگرام کے کچھ اہم فچر کے بارے میں سیکھا۔ یہ تقریباً ہر پروگرام میں استعمال ہوتے ہیں۔ آئیے اب اس کی آؤٹ پٹ دیکھتے ہیں۔ اس کو کپیل کریں گے تو یہ آؤٹ پٹ ہو گی۔

Welcome to C++ Program:

آپ اسی پروگرام کو کئی طریقوں سے لکھ سکتے ہیں مثلاً

```
# include <iostream.h>
void main( )
{
    cout <<
```

"Welcome to C++ Program:" ;

```

}
یا
void main( )
{
cout
<<
"Welcome" <<"to" <<"C++ Programe";
}

```

یہ مختلف طریقے بتاتے ہیں کہ مقصود صرف یہ ہے کہ آپ کو C++ کے بنیادی فچرز کے بارے میں معلومات حاصل ہونی چاہئے۔ ان سب پروگرامز کی اوت پٹ ایک ہی ہو گی لیکن صرف تکمیل کی مختلف ہے۔

کو منش شامل کرنا:

کسی بھی پروگرام میں کو منش شامل کرنا ایک بھروسہ پروگرام کی صفت جانی جاتی ہے۔ آپ پروگرام میں یہ کو منش صرف اپنی کہوت کے لئے شامل کرتے ہیں۔ یہ کو منش آوت پٹ کا ایک حصہ نہیں ہوتے ہیں بلکہ یہ آپ کو پروگرام کے متعلق اضافی معلومات فراہم کرتے ہیں اور جب آپ اس پروگرام کو ایڈیٹ (تبدیل) کرنا چاہئے ہیں تو آپ کے لئے مدد نامہ ہوتے ہیں یعنی آپ کو اس بات کی وضاحت کرتے ہیں کہ پروگرام کی فلاں لاں کا کیا مقصد ہے۔ آپ اپنے پروگرام میں دو طریقوں سے کو منش شامل کر سکتے ہیں۔

```

// output statement
/* output           اور
of Marks
average */

```

اب یہ پروگرام میں کسی جگہ اور کیسے لکھے جاسکتے ہیں آئیے دیکھتے ہیں۔

```

#include <iostream.h> //header file
void main() //main Program
{
    cout <<"How Are you Choudhry:";

    /*our Printing Message*/
}
/*End of
Program*/

```

آپ اس طرح اپنے پروگرام میں کو منش شامل کر سکتے ہیں۔ آپ سوچ رہے ہوں گے کہ // اور /* / میں کیا کرتی ہے؟ آپ جب بھی سنگل لائنز یعنی صرف ایک لائن کو کو منٹ تیکٹ شامل کرنا چاہئے ہیں تو یہ // استعمال کرتے ہیں جبکہ اگر آپ کو کو منٹ تیکٹ ایک لائن سے زیادہ ہے تو ہر دفعہ // علامت لکھنے کی بجائے آپ تیکٹ کے آغاز پر /* اور اختتام پر */ یہ علامات تحریر کر دیں۔ جو بھی لائن یعنی تحریر ان کے درمیان ہو گی کمپانکر اس کو Read نہیں کرتا۔

ویری ایبل:

ویری ایبل ایک ایسی علامت ہے جو کمپیوٹر کی میموری میں ڈینا محفوظ کرنے کے لئے استعمال ہوتی ہے یا ویری ایبل میموری کے ایک بکھرے کا نام ہے جو آپ C++ پروگرام میں انفارمیشن محفوظ کرنے کے لئے استعمال کرتے ہیں۔ میموری کا ہر بکھر جو آپ اپنے پروگرام میں ڈینا نہ کرتے ہیں ایک مخصوص قسم کا ڈینا محفوظ کر سکتا ہے یعنی جس ناپ کا ویری ایبل ہو گا وہ صرف اس ناپ کا ڈینا شور کروانے کے لئے استعمال ہو گا۔

نوت: یہ ناپ کیا ہے؟ آپ اس باب میں دیکھیں گے۔

مثلاً آپ ایک ویری ایبل نامیک ویلیو محفوظ کرنے کے لئے لکھتے ہیں تو پھر آپ اس میں کریکٹر یا اعشاری نظام میموری میں محفوظ نہیں کروا سکتے۔ آپ صرف نمبرز 1, 2, 3, 4, 5 کروا سکتے ہیں اور آپ جو معلومات میموری میں محفوظ کریں گے وہ ویری ایبل کی ویلیو ہو گی۔ کسی بھی ویری ایبل کو ویلیو اسی طرح آسانی کی جاتی ہے۔

variable = expression/value;

(Keywords & Identifiers):

کی ورڈز اور ایڈنٹیفائرز:

کی ورڈز کو پروگرامنگ زبان میں ریزرو ورڈز کہتے ہیں یا ایسے الفاظ ہوتے ہیں جو لینگوچ نے اپنے لئے ریزرو کئے ہوتے ہیں اور یہ خاص مقاصد کے لئے استعمال ہوتے ہیں اور پروگرام خود سے انہیں دفعہ دفعہ ڈینا نہیں کر سکتا اور نہ ان کو بطور ویری ایبل استعمال کر سکتا ہے۔ مثلاً , cout , count , int وغیرہ

ایسا نام جو آپ ویری ایبل کے لئے استعمال کرتے ہیں یا ایسا نام جو C++ میں کسی بھی اوبجیکٹ کے لئے منتخب کرتے ہیں وہ ایڈنٹیفائر کہلاتا ہے۔ ایک ایڈنٹیفائر (حروف تہجی) الفنیریک کریکٹر کا ایک سڑنگ ہوتا ہے۔ اس کا پہلا حرف ہمیشہ کریکٹر ہوتا ہے۔ تقریباً کل 53 الفا یک کریکٹر زیں جن میں 52 حروف اور ایک (-) اندر سکور ہے۔ ان کے علاوہ 10 نامیریک کریکٹر زیں جن میں کسی بھی ایڈنٹیفائر کا آغاز C++ آپریٹر (+, -, *, /) وغیرہ سے شروع نہیں کر سکتے۔ ہم نے پہلے بھی بتایا ہے کہ ایک Case sensitive SUM اور sum میں فرق ہے۔

نیوالائن کر کریکٹر:

نیوالائن کر کریکٹر کیا ہے؟ اس پروگرام کو دیکھیں اور بعد میں سمجھنے کی کوشش کریں۔

```
#include <iostream.h>
void main()
{
    cout << "Welcome Mr. Shoaib.\n";
    cout << "How are you Lala Rukh." << endl;
}
```

اس پروگرام کو جب آپ کپائل کریں گے تو اس کی آؤٹ پٹ کچھ یوں ہو گی۔

Welcome Mr. Shoaib

How are you Lala Rukh.

پہلی لائن میں \n علامت نیوالائن کر کریکٹر کہلاتی ہے۔ آپ جب بھی کسی آؤٹ پٹ سیٹ میں کسی آؤٹ پٹ کے بعد یہ کریکٹر شامل کریں گے تو یہ کپاٹر کو یہ

واضح کرے گا کہ اس کے بعد جو بھی تحریر پر نہ کرنی ہے وہ اگلی یعنی نئی لائن پر ڈیپلے ہو۔ اسی طرح یونچے والی لائن میں endl لکھا ہوا ہے۔ یہ پبلے سے ڈیفائن کیا ہوا کی ورد ہے اور یہ نئو لائن کریکٹر 'n' کا مقابل ہے یہ endl کریکٹر کہلاتا ہے اور یہ بھی نئی لائن کے لئے استعمال ہوتا ہے۔

ڈیٹا ناپس:

ہم نے اوپر بیان کیا ہے کہ آپ ایک ویری اینڈل میں صرف اسی قسم کا ڈیٹا شور کر سکتے ہیں۔ یعنی اس کی ویلوواسی ٹائپ کی ہو گی جس ٹائپ کا وہ ویری اینڈل ہو گا۔ اب یہ ٹائپ کیا ہے؟ ٹائپ میں یہ بات واضح ہوتی ہے کہ آپ کا ویری اینڈل کس قسم کا ہے۔ مثلاً

```
int cubes;
```

اس میں cubes ویری اینڈل کا نام ہے جبکہ سکی کالن (:) ٹائمٹ میٹ کا اختتام ہے اور int ویری اینڈل کی ٹائپ ہے۔ یونچے ہم نے ایک ٹیبل بنایا ہے جس میں C++ کی بنیادی ناپس کے متعلق معلومات ہے۔

ویری اینڈل ٹائپس C++

ٹیبل 1.1

Table

Keyword	Low	High	Bytes of Memory
char	-127	127	1
short	-32,767	32,767	2
int	-214,748,3687	214,748,3687	4 or 2
long	-2,147,483,687	2,147,483,687	4
float	3.4×10^{-98}	3.4×10^{38}	4
double	1.7×10^{-308}	1.7×10^{308}	8
long double	1.7×10^{-4932}	1.7×10^{4932}	10

نوت: int ٹیکس پر 4 بائنس ریز روکتا ہے۔ ویسے اس کا سائز 2 بائنس بھی ہوتا ہے۔

ان ڈیٹا ناپس کو استعمال کرتے ہوئے آپ ان +ve ve دو نوع نویست کی ویلوواز کو محفوظ کر سکتے ہیں۔ یعنی آپ صرف +ve ve ویلوواپنے ویری اینڈل کو آسائیں کرنا چاہتے ہیں تو اس کے لئے ان آسائند (Unsigned) ڈیٹا ناپس استعمال کی جاتی ہیں اور ان کی رنچ بھی تبدیل ہو جاتی ہے۔ یونچے ان ناپس کا بھی ایک ٹیبل درج ہے۔

ان سائند (Unsigned) ڈیٹا ناپس C++

ٹیبل 1.2

Table

Keyword	Low	High	Bytes of Memory
unsigned char	0	255	1
unsigned short	0	65,535	2

unsigned int	0	4,294,967,295	4
unsigned long	0	4,294,976,295	4

ان ڈیٹا نیچس کی مدد سے آپ ویری اینڈر کو مطلوبہ دیلوں آسانی کر سکتے ہیں۔ ان ڈیٹا نیچس کو آپ کس طرح استعمال کر سکتے ہیں۔ آپ آگے اس باب میں تفصیل سے پڑھیں گے۔

ڈیٹا نیچس: integer

ایک integer کی میں کم نمبر ہوتا ہے مثلاً -3, -2, 1, 2, 3, -1، وغیرہ۔ اب آئیے دیکھتے ہیں کہ اس قسم کی نیچس میں کون کون سی ڈیٹا نیچس آپ استعمال کر سکتے ہیں۔

byte , short , int , long

نوت: int کی 32768 سے 32768+1 بائس ہوتی ہے۔

آپ جب بھی integer ویلو میوری میں موکرنا چاہتے ہیں تو ان میں سے کوئی بھی ڈیٹا نیچس کے ویری اینڈر کو آپ یوں ڈیلکٹ کر سکتے ہیں۔ اس کے علاوہ اس کی

byte	small;
short	medium;
int	number;
long	bigral;

یہ اور ویری اینڈر ڈیلکٹ کرنے کا طریقہ ہے لیکن کہ number کی ڈیٹا نیچس ایسا ہے اب int number ویری اینڈر کا ہے اور آپ اس کو بھی int نیچس کا ڈیٹا آسانی کر سکتے ہیں۔ اب جب بھی یہ شیٹ میٹ کپائل ہو گی تو نیجے دل میں وہی کے لئے دو بائس ریزو کر دی جائیں گی۔ اب اس کے بعد مرحلہ ویری اینڈر اینی شلا بیز کر کر دیں۔

int number = 81;

یا اس کے علاوہ ایک طریقہ یہ بھی ہے۔

```
int number, a;
number = 10;
a = 101;
```

اسی طرح آپ کسی بھی نیچس کا ویری اینڈر اینی شلا بیز کر سکتے ہیں۔

```
long    bigral;
bigral = a+number;
bigral = 10131
```

آئیے ایک پروگرام دیکھتے ہیں جس میں ویری اینڈر کا استعمال کیا ہوا ہے۔

مثال نمبر 1.2

```
# include <iostream.h>
```

```
//This program shows initialisation of variables.
```

```

void main( )
{
    int a;      //Declaration
    short b;
    long sum;
    a = 10;     //initialisation
    b = 7;
    Sum = a+b;
    cout <<"The Sum of:" <<a <<"and" <<b
           <<"is" <<"Sum";
}

```

اب اس پروگرام میں یہ ہو رہا ہے کہ سبھے تین ویری ایبلز int جس کی ناپ ہے اور b کی ناپ short ہے اور long کا ایک ناپ کا sum ہے۔ بعد میں a اور b کو دیلو آسانی سے یعنی شاید کیا ہے اور بعد میں sum میں a + b کروایا ہے یعنی ان دونوں کا رزلٹ sum میں سور ہو گا۔ جب آپ اس پروگرام کو کپاٹ کریں تو اس کی آٹھ پٹ یہ ہو گی۔

The Sum of: 10 and 7 is: 17

کریکٹرڈیٹا ناپ:

C++ میں کریکٹرڈیٹا ناپ کو سور کرنے کے لئے char کی ناپ استعمال کی جاتی ہے۔ اس میں کریکٹر کی ہر دلیلو کے گرد'A' (سنگل کو ماکوٹشن) لگاتے ہیں۔ یہ کریکٹر کو سڑنگ سے مختلف ظاہر کرنے کے لئے لگایا جاتا ہے۔ یہ میوری میں 8 bits یعنی 1 باہیت جگہ گھیرتا ہے۔ یہ integer کا ایک حصہ ہے اس لئے آپ کریکٹر میں نریک دلیلو بھی تحریر کر سکتے ہیں۔ مثلاً

char d = 10;

char b = 'A';

char سے مراد کریکٹر ہے آپ جب بھی char کی ناپ لکھتے ہیں تو اس ناپ کا ویری ایبل بطور کریکٹر پرست کر جاتا ہے۔ اور سسٹم خود بخود اس کا ASCII کوڈ میوری میں سور کرتا ہے۔ آئیے اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 1.3

```

# include <iostream.h>
# include <conio.h>
void main( )
{
    char variable = 'Z';
    char a = 41;
    cout <<"output is:" <<a <<endl
           <<"and" <<variable;
}

```

```
getch();
}
```

اس طرح سے آپ اپنے پروگرام میں کریکٹ استعمال کر سکتے ہیں۔ آپ حیران ہوں گے کہ اس پروگرام میں ایک لائن کا اضافہ ہو گیا ہے کیا ہے؟ یہ C++ کا پہلے سے بنا ہوا ایک فنکشن ہے۔ اس کا فائدہ یہ ہے کہ جب آپ پروگرام کی آڈٹ پٹ دیکھنے کے لئے اسے کپیاں کریں گے تو وہ فوراً اپنے C++ سورس کوڈ پر آ جاتا ہے لیکن اس کے لکھنے سے وہ اس وقت تک سورس وندوز میں واپس نہیں آئے گا جب تک آپ کی بورڈ سے کوئی کی (Key) پر لس نہیں کریں گے۔ get character کا مطلب ہے یعنی ایک کریکٹ پر لس کریں کریں لیکن اس سے متعلق معلومات میں سورس نہیں ہیں۔ اس کے لئے آپ کو (conio.h) اسے مانند استعمال کرنی پڑتی ہے۔

فلونگ پوائٹ ڈیٹا نامہ

ایسی ولیوز جن میں آپ نے اعشاری نظام استعمال کیا ہو اور وہ integer نامہ میں استعمال نہیں کی جاسکتی ہوں وہ فلوونگ پوائٹ ڈیٹا نامہ میں محفوظ کی جاتی ہیں۔ مثلًا 3.14 ، 11.561 ، 11.56 وغیرہ۔ اس کے لئے آپ float اور double استعمال کرتے ہیں۔

مثال نمبر 1.4

```
float a;
float a = 3.14;

//floating point variables
void main( )
{
    double result;
    float a = 3.14;
    double d = 5.69;
    result = a*d;
    cout << "Answer is:" <<result;
    getch( );
}
```

اس کو ایک مثال سے دیکھتے ہیں۔

حسابی (Arithmetic) آپریٹرز:

ایک آپریٹر ایسی علامت ہوتی ہے جو ایک یا ایک سے زائد ایک پریٹ کر کے ایک ویری اسٹبل کو اس کی ولیو آسان کرتی ہے۔ آپ نے اس کتاب میں آڈٹ پٹ آپریٹر کے بارے میں پہلے ہی پڑھا ہے اس کے علاوہ ایک اہم آپریٹر آسانٹ (=) ہے جو ایک ولیو ویری اسٹبل کو آسان کرنے کے لئے استعمال ہوتا ہے۔ یہ بائیس طرف والی ولیو واریس طرف کے ویری اسٹبل کو آسان کرتا ہے۔ اس کے علاوہ چھ اہم integer حسابی آپریٹر ہیں۔

Table

کوونکہ $2+3*1 = 32$ not 50 اس لئے اگر % پر پہلی توجیہ میں نہیں ملے تو اس کا پہلی توجیہ کا انتظام ہے۔

پہلی توجیہ

مثال	وضاحت	آپریٹر
$a+b$	جمع	+
$a-b$	تفريق	-
$a*b$	ضرب	*
a/b	تقسيم	/
$a \% b$	باقیا	%
$-a$	منفی	-

ان آپریٹر کو C++ میں یہ سہالا کیا جاسکتا ہے؟ اس کی ایک مثال پیچے درج ہے۔

مثال نمبر 1.5 آپریٹر کا استعمال

//use of Arithmetic operators

```
void main( )
{
    int a=5, b=3;
    cout << a << "+" << b << "=" << (a+b) << endl;
    cout << a << "-" << b << "=" << (a-b) << endl;
    cout << a << "*" << b << "=" << (a*b) << endl;
    cout << a << "/" << b << "=" << (a/b) << endl;
    cout << a << "%" << b << "=" << (a%b) << endl;
    getch();
}
```

اس پروگرام میں ہم نے پیچے والے پروگرام کی نسبت قدر مختلف طریقے سے رزلٹ پرنت کروایا ہے تاکہ آپ C++ پروگرامنگ کے تمام فنچورز کے بارے میں معلومات حاصل ہو جائے۔ آپ اس کے علاوہ a اور b کا رزلٹ کسی دیری ایبل میں بھی محفوظ کر سکتے ہیں۔ مثلاً

$$c = a+b$$

$$c = a \% b \quad \text{یا}$$

اس کے علاوہ آپ نے دیکھا کہ اور ہم نے ہیڈرفائل iostream.h نہیں لکھیں وہ آپ خود لکھیں گے۔ اس پروگرام کی آڈٹ پٹ یہ ہو گی۔

$$a+b = 8$$

$$a-b = 2$$

$$a\times b = 15$$

$$a/b = 1$$

$$a/b = 2$$

اس میں a/b پر غور کریں اس کا رات 1 ہے اس کی وجہ یہ ہے کہ اس کا رات اعشاری نظام میں آتا ہے لیکن ہم نے ان کو int ڈیکلائر کیا ہے۔ اس لئے وہ اعشاری نظام کو نظر انداز کر دے گا۔ اس کے لئے آپ کو تاپ کنورٹ کرنا ہو گی جو بعد میں آپ پڑھیں گے۔

آپ پریز کی فوکیت:

ایک ایک پریش میں ایک سے زیادہ آپ پریز بھی موجود ہو سکتے ہیں۔ لہذا ہمیں اس بات کے بارے میں معلومات ہونی پڑے کہ ایک پریش میں کون سا آپ پریز سے پہلے پر فارم ہو گا۔ آپ پریز اس طرح Evaluate میں آپ پریز اس ترتیب سے حل کئے جاتے ہیں۔

Highest to lowest

() []
* - Multiplication

/ - Division

% - Remainder / Modulus

+ - Addition

- - Subtraction

<<

=

$$\begin{array}{r} \frac{1}{4} \\ 16 \\ \hline 4 \\ \text{Remainder} \leftarrow 12 \end{array}$$

یوزی آپ پریز:

C C++ لینکوچ کی طرح ++ اور -- آپ پریز بھی استعمال کرتی ہے۔ ان میں ++ کو اضافی اور -- کو اضافی اور -- آپ پریز کہتے ہیں۔ یعنی آپ پریز ان میں ++ اپنے آپ پریز میں ایک اضافہ کرتا ہے جبکہ -- اپنے آپ پریز میں ایک کم کرتا ہے کیونکہ ان آپ پریز کا آپ پریز صرف ایک ہوتا ہے اس لئے انہیں یوزی آپ پریز بھی کہتے ہیں۔ مثلاً

$a++;$ یا $++a;$

$a--;$ یا $--a;$

یعنی آپ یہ آپ پریز ویری اسٹبل کے دونوں طرف لگاتے ہیں لیکن دونوں صورتوں میں رات مختلف ہو گا۔ اس کا وہ فتح اور پری فتح کہتے ہیں۔ پوسٹ فتح میں آپ پریز کی ویبو حاصل کرنے کے بعد اس میں ایک کا اضافہ ہوتا ہے جبکہ پری فتح میں پہلے ایک کا اضافہ ہوتا ہے اور بعد میں آپ پریز کی ویبو حاصل کی جاتی ہے۔ اس کو آپ مثال سے آسانی سمجھ لیں گے۔ پہلے ہم لے ایک سادہ کوڈ لکھا ہے۔

مثال نمبر 1.6

```
{
int a=7, b=8;
++a;
--b;
cout <<"a=" <<a <<", b=" <<b <<endl;
a++;
}
```

```

    b--;
    cout << "a=" <<a <<" ,b=" <<b;
    getch();
}

```

جب آپ اس پروگرام کو ایگزکیوٹ کریں گے تو یہ آڈٹ پٹ ڈسپلے ہو گی۔

a=8, b=7

a=9, b=6

آئیے اب نکلیں اس پروگرام لکھتے ہیں جو اس کی مکمل وضاحت کرے گا۔

مثال نمبر 1.7

```

#include <conio.h>
#include <iostream.h>
void main()
{
    int a=11, b;
    b=++a;
    cout << "a=" <<a <<" ,b=" <<b << endl;
    b=--a;
    cout << "a=" <<a <<" ,b=" <<b << endl;
    b=a--;
    cout << "a=" <<a <<" ,b=" <<b << endl;
    getch();
}

```

اب اس پروگرام کو آپ ناٹپ کریں اور ایگزکیوٹ کریں۔ اس کا رزالٹ یہ ہو گا۔

a=12, b=12

a=11, b=11

a=11, b=10

آرٹھمیٹیک آسانمنٹ آپریٹرز:

C++ میں آپ ایک کوڈ کوئی طریقوں سے لکھ سکتے ہیں۔ اسی طرح ان میں سے ایک آرٹھمیٹیک آسانمنٹ آپریٹر ہے جس کی مدد سے آپ آپریٹرز شارٹ کٹ طریقے سے استعمال کر سکتے ہیں۔ اس سے کوڈ آسان ہو جاتا ہے۔ کوڈ کم ناٹپ کرنے کی ضرورت ہوتی ہے جس سے نامنفع جاتا ہے۔ مثلاً

a = a+4;

اس سٹیٹ میٹ کو آپ یوں بھی لکھ سکتے ہیں۔

$$a+ = 4;$$

اس سینٹ میٹ میں آپ a ویری ایمیل میں 4 جمع کرنے کے بعد نئی ولیو a میں سور کر رہے ہیں اور یہی کام اور پر والی سینٹ میٹ میں ہو رہا ہے۔ آئیے اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 1.8 ارجمند آپ پریز

```
# include <iostream.h>
# include <conio.h>
void main( )
{
    int result = 5;
    int answer = 7;
    result+=5;//equivalent to result=result+5;
    cout <<"Result is:" <<result <<endl;
    answer%=2;//equalent to answer=answer%2;
    cout <<"Answer is:" <<answer;
    getch( );
}
```

اب اس پروگرام کو ایگزیکیوٹ کریں۔ اس کی آؤٹ پٹ بالکل دیے ہی ہو گی۔ جنہیں آج تکمیک آپ پریز سے آپ پروگرام حل کرتے ہیں۔

آؤٹ پٹ

Result is : 10

Answer is : 1

ریٹن آپ پریز:

یہ آپ پریز دو ولیوں کا موازنہ کرنے کے لئے استعمال ہوتے ہیں۔ یہ ولیوں C++ کی ڈائٹا نچس میں سے ہوئی چاہئے۔ اس کے علاوہ یہ آپ پریز دونوں آپ بینڈ کے درمیان تعلق کو ظاہر کرتے ہیں یعنی یہ دونوں آپ بینڈ کی برابری پر ان کا آرڈر واضح کرتے ہیں۔ ان آپ پریز کا نیبیل نیچو درج ہے۔

وضاحت	آپ پریز
a==b;	Equal to برابر ==
a!=b;	Not equal to برابر نہیں !=
a<b	Less than چھوٹا ہے <
a>b	Greater than بڑا ہے >
a<=b	Less than or equal to چھوٹا ہے یا برابر <=
a>=b	Greater than or equal to بڑا ہے یا برابر >=

یہ آپ میرزا ہم لوپس یا اس کی کندیش میں استعمال کرتے ہیں۔ جو آپ اگلے باب میں پڑھیں گے۔ یعنی کہ اگر یہ اس کے برابر ہے تو یہ آؤٹ پٹ ہونی چاہئے ورنہ کوئی اور آؤٹ پٹ ہوگی۔ لیکن آئینے دیکھتے ہیں کہ ان کو کیسے استعمال کیا جاسکتا ہے۔

مثال نمبر 1.10 آپ میرزا

```

int num = 5;
cout << "sum is less 10;" << (num<10) << endl;
cout << "sum is equal to 10;" << (num==10) << endl;
cout << "sum is greater 10;" << (num>10) << endl;
getch();
}

```

اب اس پروگرام کو ذرا سمجھنے کی کوشش کرو۔ اس میں ایک دیری ایبل num ہے جس کی ولیوو 5 ہے۔ اس کے بعد یہ کندیش ہے کہ کیا یہ نمبر 10 کے برابر ہے یا اس سے بڑا ہے اور یا پھر چھوٹا ہے۔ میرزا 10 سے چھوٹا ہے اس لئے جہاں چھوٹا ہے (>)۔ کندیش ہو گی اس کی آؤٹ پٹ 1 ہو گی باقی کا رزلٹ 0 ہو گا۔ اس سے ظاہر ہوا کہ جب بھی ہماری لندیش ہے ورنہ کما موازنہ درست ہو گا تو رزلٹ 1 شو ہو گا ورنہ صفر ڈپلے ہو گا۔

ٹائپ کنورٹن:

ٹائپ کنورٹن کی ضرورت اس وقت پڑیں آتی ہے جب آپ میرزا ٹائپ کی ولیوو کی دوسرا مختلف قسم کی ڈیٹائپ کے دیری ایبل کو آسانی کرنا چاہتے ہوں۔ لیکن آپ کو C++ میں اتنا زیادہ فکر مند ہونے کی ضرورت نہیں ہے بلکہ C++ میں عام طور پر یہ خود بخوبی ہو جاتا ہے۔ مثلاً

```

int c=7;
float a=3.142;
double res=c*a;
cout << "Answer is :" << res;

```

اب اس پروگرام کو اگر آپ ایگر یکیوٹ کریں گے تو اس میں کوئی ایرٹیس ہو گا کیونکہ کپاٹر کو معلوم نہ ہے آپ c کو a سے ضرب دینا چاہتے ہیں اور وہ خود بخوبی کی ولیوو 0 کی ولیوو 7 تصور کرے گا۔

اس کے علاوہ اگر آپ ٹائپ کا سنتگ کرنا چاہتے ہیں تو یہ بہت آسان ہے۔ اس لئے پہلے آپ کو یہ طریقہ اختیار کرنا ہو گا۔

variable = int(expr);

اس شیٹ میں expr کی ولیوو int میں کنورٹ ہو گی اور دیری ایبل کو آسان کر دی جائے گی لیکن فرض کریں کہ آپ کا پہلا نمبر یعنی exp میں اعشاری نمبر ہے یعنی 4.19 ہے تو یہ اس کا اعشاریہ اور بعد والی ولیوو ختم کر دے گا اور صرف 4 دیری ایبل کو آسان ہو گا۔ آئینے اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 1.10 ٹائپ کا سنتگ پروگرام

```
void main( )
{

```

```
    double d=3415.3124;
    int i=int(d);
}
```

```

cout << "d" << d;
cout << endl << "i=" << i;
getch();
}

```

اب اس پروگرام میں یہ ہو رہا ہے کہ ڈبل ویری اینٹل d کی دلیلوں کو int تاپ کے ویری اینٹل میں سورکھیا گیا ہے۔ d کی تاپ کا سٹنگ بھی کی ہے۔ اس سے d کا اعشاریہ ختم ہو جائے گا یوں اس پروگرام کی آٹھ پٹ ہو گی۔

d = 315.3124

i = 3415

ایک بات ذہن نشین کر لیں کہ جب چھوٹی تاپ کا ویری اینٹل بڑی تاپ کے ویری اینٹل میں کورٹ کیا جاتا ہے تو کسی آپ پر یہ کی ضرورت نہیں ہوتی۔

ان پٹ:

آپ نے اب سے پہلے ویری اینٹل ڈیلیٹ کرنا سیکھا اور یہ زان کو خود آسانی کرتے تھے لیکن آپ اس کے علاوہ یہ ان پٹ رن نام پر یوزر سے بھی لے سکتے ہیں اور یہ کوئی مشکل کام بھی نہیں ہے۔ اس کے لئے آپ cin سریم استعمال کر سکتے ہیں اور یہ console ان پٹ کا مخفف ہے۔ اب اس کو کب اور کیسے استعمال کرنا ہے آئیے دیکھتے ہیں۔

مثال نمبر 1.11 ان پٹ پروگرام

```

void main()
{
    int number;
    char ch;
    cout << "Enter a number:" ;
    cin >> number;
    cout << "Enter a Character:" ;
    cin >> ch;
    cout << "number=" << number << "character" << ch;
    getch();
}

```

اس میں ہم نے cin کے بعد << آپ پر یہ استعمال کیا ہے۔ اس کو ان پٹ Extraction آپ پر بھی کہتے ہیں۔ یہ cin سریم کے ساتھ کی بورڈ سے ان پٹ حاصل کرنے کے لئے استعمال کیا جاتا ہے۔ اس میں پروگرام کو جب آپ ایگزیکیوٹ کریں گے تو پہلی لائن پرنٹ ہونے کے بعد سٹرم کر جائے گا اور جیسے ہی آپ کوئی نمبر درج کریں گے وہ ویری اینٹل کو آسان کر دیا جائے اور پروگرام یا سٹرم دوسرا ان پٹ مانگے اور بعد میں اس کا رازٹ شو کروا دے گا۔ آپ اس کے علاوہ ایک لائن میں ایک سے زائد ان پٹس بھی لے سکتے ہیں۔ مثلاً

char ch, ch1;

visit our website: www.papersbook.Blogspot.com

```
cin >> ch >> ch1;
```

سڑک ان پٹ:

اوپر آپ نے cin کی مدد سے ان پٹ لینے کا طریقہ سیکھا ہے لیکن cin کے ساتھ ایک پرالام ہے کہ اس کی مدد سے آپ سڑک ان پٹ اُن طریقے سے نہیں لے سکتے ہیں۔ اس میں لا جیکل امیر ہوتا ہے۔ یہ دراصل صرف نریک دبلیو ز کے لئے بہترین ہے لیکن آپ ایک سڑک لکھنا چاہتے ہیں۔ جس میں کافی جگہ (پسیں) ہو تو cin اس میں لا جیکل امیر پیدا کرتا ہے۔ یہ پسیں کو ایک الگ کریکٹ سڑک تصور کرتا ہے۔ اس کے لئے آپ C++ کا اہم فچر (gets) استعمال کر سکتے ہیں۔ gets اس میں سڑک کا مخفف ہے اب اس کو پروگرام میں لکھ سکتے ہیں۔

```
cout << "Enter your name:";
```

```
gets(name);
```

آپ کا نامی اسٹریم ہے۔ اس کے لئے آپ کو ایک اور کام کرنا ہو گا کہ ڈائریکٹری شامل کرنا ضروری ہے لیکن اس کی اضافی معلومات name میں نہیں ہے اس کے لئے * <stdio.h> ہیئت رفائل استعمال ہوئی ہے۔

```
# include <stdio.h>
```

اس کتاب کے حوالہ سے ایک بات بہت ایک ہے کہ ہم نے اس میں ہر مثال میں ہیئت رفائل شامل نہیں کیس وہ آپ خود لکھیں گے اور اسکے main() میختہز میں () اور آخر پر () ضرور لکھا کر لیں۔ clrscr() سے آپ کی آؤٹ پٹ سکرین صاف ہو جاتی ہے۔ () آؤٹ پٹ سکرین کو اس وقت ختم نہیں کرتا جب تک کہ آپ کی یورڈ سے کمی کی نہ رہے۔ اگر آپ کو کسی کی ورثی یا پہلے سے بنے ہوئے فونکشن کی ہیئت رفائل کا علم نہیں ہے تو کسر اس کے نیچے لا کیں اور اس سے متعلق تمام معلومات کھل جائیں گی۔

مشق

سوال نمبر 1: مختصر جواب دیں۔

- (i) C++ پروگرام ایگزیکوٹ ہونے سے پہلے کن مرحلہ میں سے گزرتا ہے؟
 (ii) C++ میں دو ڈائناپس کون کون سی ہیں؟
 (iii) cin اور cout میں کیا فرق ہے؟

سوال نمبر 2: C++ کا سب سے پھرطی ایک کیوٹیبل پروگرام لکھیں (خواہ وہ کچھ آؤٹ پٹ دے یا نہ دے لیکن کسی ایر کے بغیر ایگزیکوٹ ہو)

سوال نمبر 3: ایک پروگرام لکھیں جو یوزر سے قلن میں بطور ان پٹ لے پھر ان کا مجموعہ، اوست اور تینوں نمبرز کی پراؤٹ معلوم کریں۔
 نوٹ: آپ نے وہی تکنیک استعمال کی ہے جو اس باب میں پڑھی ہے۔

سوال نمبر 4: اس مساوات کو حل کریں۔
$$Z = ax^3 + 3$$

 فرض کریں کہ a کی ولیو 4 اور x کی ولیو 2 ہے۔

سوال نمبر 5: فرض کریں کہ ایک کیوب فٹ میں 3.41 گیلن آتے ہیں اور آپ یوزر سے گیلن کی ولیو انگلیں یعنی یوزر سے ان پٹ لیں اور پھر اس کے مساوی کیوب فٹ میں ولیو شو کروائیں۔

سوال نمبر 6: ایک ایسا پروگرام لکھیں جو یہ آؤٹ پٹ شو کروائے۔

7

14

15

نوٹ: اس میں آپ نے $a = 7$ کو کانسٹنٹ ڈیکلر کرتا ہے۔

جوابات

1: جواب

C++ پر ڈراما ایگز کیوٹ ہونے سے پہلے مندرجہ ذیل چھڑاں میں سے گزرتا ہے۔

پری پر اس	(ii)	ایڈٹ	(i)
انک	(iv)	کپائل	(iii)
ایگز کیوٹ	(vi)	لوڑ	(v)

C++ کی اہم ڈھانچائیں مندرجہ ذیل ہیں۔

long double, double, float, long, int, short, char

unsigned long, unsigned int, unsigned short, unsigned char

cout<< کا آٹھ پت ایجاد کرنے پر یہ insertion ہے اور جب بھی آپ کسی لائن کی آٹھ پت دیکھنا چاہتے ہیں تو وہ cout میں لکھتے ہیں۔

: cin>> کا ان پت آپ پر یہ ہے لیکن یہ یمنہ عالمان پت لینے کے لئے استعمال ہوتا ہے اور یہ console input کا مخفف ہے۔

2: جواب

C++ کا سب سے چھوٹا ایگز کیوٹ بیل ڈرام یہ ہے۔

```
void main(void)
{
}
```

3: جواب

```
void main(void)
{
    clrscr();
    int a,b,c, sum, pro;
    float avg;
    cout <<"Enter Three integer values:" ;
    cin>> a>> b>> c;
    sum = a+b+c;
    pro = a*b*c;
    avg = float(sum)/3 //type casting
    cout <<"\n Sum=" <<sum;
```

```

cout << "\n Product=" << pro;
cout << "\n Average=" << avg;
getch( );
}

```

$$Z = ax^3 + 3$$

جواب : 4

```

void main(void)
{
    clrscr();
    cout << "\n Solving Z=ax^3+3:" ;
    int a=4, x=2;
    int Z=0;
    Z = a * (x*x*x) + 3;
    cout << "\n Answer=" << Z;
    getch();
}

```

جواب : 5

```

void main(void)
{
    clrscr();
    int gallons;
    float ans;
    cout << "\n Enter value of gallons:" ;
    cin >> gallons;
    ans = gallons / 3.41;
    cout << "\n gallons << in cubic feet:" << ans;
    getch();
}

```

جواب : 6

```

void main(void)
{
    clrscr();
}

```

```
const int a=>;
int ans;
cout <<ans <<endl;
ans = a+a;
cout <<ans <<endl;
ans++;
cout <<ans <<endl;
ans-=2
cout <<ans;
getch();
}
```

Bsit past papers and books

باب نمبر 2

کنٹرول سٹرکچر اینڈ فنکشنز

اکثر آپ نے دیکھا ہے کہ کام ایک ہی فلو میں شروع سے آخر تک ختم نہیں ہوتے بلکہ آپ اس کا کچھ حصہ پہلے اور کچھ بعد میں کامل کرتے ہیں۔ اسی طرح زیادہ طرح پروگرام بھی شروع سے آخر تک ایک ہی آرڈر میں ایگزیکوٹ نہیں ہوتے۔ بعض اوقات ایک یکیوں کنٹرول پروگرام کے ایک حصے سے دوسرے کوٹر اسٹر کر دیا جاتا ہے۔ اسی سیٹ میٹس جو کنٹرول ایک لائن سے دوسری کو دیتی ہیں وہ کنٹرول سٹرکچر یا کنٹرول سیٹ میٹس کہلاتی ہیں۔ یہ مندرجہ ذیل ہیں۔

لوپس کنٹرول سیٹ میٹس

فنکشن ایک پروگرام کی کئی سیٹ میٹس کو ایک یونٹ میں اکھا کرتا ہے اور ان سیٹ میٹس کو ایک مخصوص نام دیتا ہے اور آپ پھر یہ گروپ پروگرام کے کسی بھی حصے سے صرف گروپ کا نام کاں کاں کر سکتے ہیں۔ فنکشن تھماں کرنے کا ایک فائدہ یہ ہے کہ آپ کے پروگرام کا سائز کم ہو جاتا ہے۔ پروگرام کوڈ تھواڑا ہو گا تو آپ کا نام بھی بچے گا اور کوڈ سمجھنا بھی آسان ہو گا۔ فنکشن کوڈ ہمیوری میں صرف ایک جگہ محفوظ ہوتا ہے خواہ یہ فنکشن کئی دفعہ ہی کاپی کیوں نہ کیا گیا ہو۔ اسی باب کے پہلے حصے میں آپ کنٹرول سیٹ میٹس اور لوپس کا درج ہے۔ میں پڑھیں گے جبکہ اس باب کے دوسرے حصے میں آپ کو فنکشن کے بارے میں بتایا گیا ہے۔ یہ باب بڑی اہمیت کا حامل ہے۔ اس میں آپ یہ تاپس پڑھیں گے۔

سیٹ میٹ continue goto کانٹرول ایڈجیٹ سینڈر رُٹیکس فنکشن یوزر دیفائیل فنکشن ریٹن ناپ فنکشن پیرا میٹر لسٹ پیرا میٹر بائی ریفرنس ریکریٹن فنکشن اور لوڈ گک ڈیفائل آر گومنٹس مشق	کنٹرول سیٹ میٹ if else-if Nested-if switch لاجیکل آپریٹرز ویری اینبل سکوپ لوپس for Nested-for while do-while بریک سیٹ میٹ
--	--

کنٹرول شیٹ میٹ:

کسی بھی قسم کا فیصلہ کرنا یا کسی بھی چیز کا انتخاب کرنا آپ کے پروگرام کا ایک بنیادی حصہ ہوتا ہے۔ مثلاً اس نوعیت کے فیصلے آپ اپنے پروگرام میں کرتے ہیں کہ اگر یوزر کا نام اور پاس ورڈ درست ہے تو وہ پروگرام میں داخل ہو گا اور نہ پہلے درست پاس ورڈ تحریر کرے۔ اگر استعمال کنندہ کے اکاؤنٹ میں پہلے ہیں تو وہ شاپنگ کر سکتا ہے ورنہ نہیں۔ اسی طرح سٹوڈنٹس کی اوسمی اور ان کا گریڈ وغیرہ معلوم کرنا۔ پروگرامنگ میں آپ ایسا کرنے کے لئے ویری اسٹبلو، کالٹنٹس اور ایکسپریشنز کو چیک کرتے ہیں۔ C++ میں اس کے لئے دو اقسام کی شیٹ میٹ میٹس استعمال کی جاتی ہیں۔

- (i) The if Statement
- (ii) The else-if statement

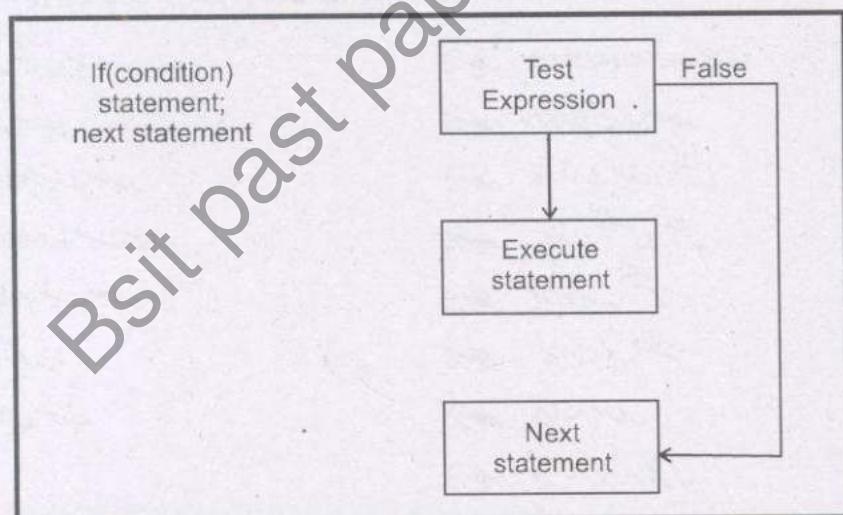
if شیٹ میٹ:

اس شیٹ میٹ کی مدد سے آپ ایکسپریشن چیک کرتے ہیں کہ کیا وہ درست ہے یا نہیں؟ اور اگر درست ہے تو کیا رزلٹ شو کروائے گی۔ اس کا جزو سینٹکس (syntax) یہ ہے۔

```
if (expression/condition)
```

```
statement;
```

اب یا آپ کی کندھیں کسی بھی نوعیت کی ہو سکتی ہے بلکہ وہ درست ہو گی تو یہ شیٹ میٹ ایگزیکیوٹ ہو گی ورنہ یا ایگزیکیوٹ نہیں ہو گی۔ اس کی حرآرکی سے آپ یہ بات سمجھ سکتے ہیں۔



اس میں آپ آسانی سے سمجھ سکتے ہیں کہ اگر ایکسپریشن True (درست) ہو گی تو اس کے فوراً بعد والی شیٹ میٹ ایگزیکیوٹ ہو گی اور اگر ایکسپریشن / کندھیں false (غلط) ہو گی تو باہر والی شیٹ میٹ ایگزیکیوٹ ہو گی۔ اس صورت میں ہماری if کندھیں میں صرف ایک شیٹ میٹ ہے لیکن اس کے بعد ایک سے زائد شیٹ میٹس بھی لکھی جاسکتی ہیں۔ اس کے لئے {} برکیٹس کا استعمال کیا جاتا ہے۔ مثلاً

```
if (condition)
```

```

statement 1;
statement 2;
:
statement n;
}

```

اپر ہم نے جآر کی میں بریکس استعمال نہیں کی اس لئے یہاں کندیش درست ہونے کی صورت میں صرف پہلی شیٹ میٹ اس کندیش کے تحت ایگز کیوٹ ہو گی جبکہ اس طریقہ کار میں بریکس میں تمام شیٹ میٹ ایگز کیوٹ ہوں گی۔ آئیے اب اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 2.1 if شیٹ میٹ کا استعمال

```

viiod main( )
{
    int x,y;
    x = 15;
    y = 20;
    if(x<y) //x is less than y then.
        cout <<"x is less than y";
    y = y-3;
    if(x==y) //equals to y then.
    {
        cout <<"x is equal to y" <<endl;
        cout <<"if statement example";
    }
    cout <<"\n End of Program.";
    getch();
}

```

اس پروگرام میں int تائپ کے دو ویری اسٹرلر x اور y ڈیکلیر اور انشا نہ کروائے گئے ہیں اور ان کی مدد سے کندیش چیک کی گئی ہے کہ اگر x ویری اسٹرلر اسے چھوٹا ہے تو یہ ڈسپلے کرے اور بعد میں y میں سے 3 تفریق کیا ہے اور یہ کندیش لگاتی ہے کہ کیا x اور y دونوں برابر ہیں۔ اب اس پروگرام کو ایگز کیوٹ کریں اور ایک بات کا خیال رکھیں کہ ہیدر فالکلر آپ خود لگائیں گے۔ اس کی آؤٹ پٹ یہ ہو گی۔

x is less than y

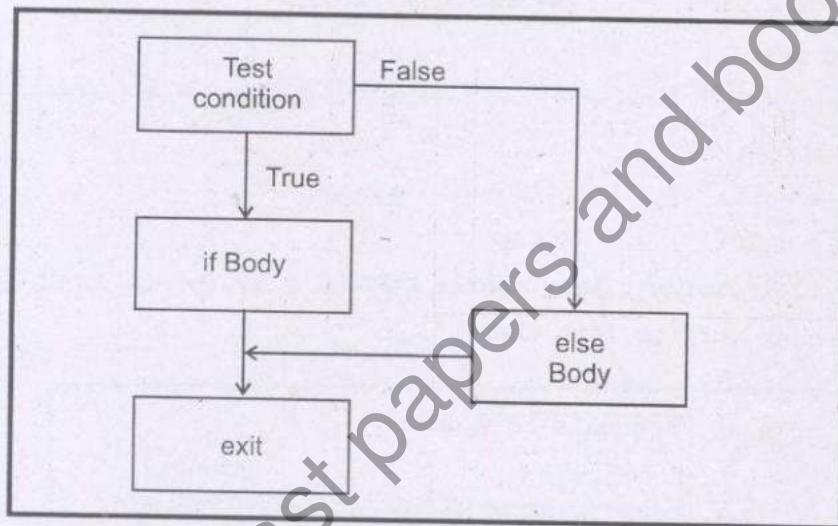
End of Program.

else-if شیٹ میٹ:

آپ نے یہ پہلی شیٹ میٹ کے بارے میں پڑھا ہے اس کے لئے آپ اسے شدید مدد بھیج رہے ہیں کہ کتنے

کرتی ہے۔ اس کی مدد سے آپ اف کنڈیشن کے علاوہ ایک اور تبادل کنڈیشن چیک کر سکتے ہیں۔ آئیے اس کی حرآرکی سے اس کی اہمیت بحث ہے۔

```
if (condition)
{
    statement;
    statement 1;
}
else
    statement;
```



حرآرکی else-if

آئیے اسی کی ایک سادہ اور آسانی مثال لکھتے ہیں۔

مثال نمبر 2.2 else-if کا استعمال

```
viiod main()
```

```
{
    int num1,num2;
    cout << "Enter two integers.\n";
    cin >>num1 >>num2;
    if(num1%num2==0)
        cout <<"num1 <<"is divisible by" <<num2;
    else
        cout <<num <<"is not divisible by <<num2;
```

```

    getch();
}

```

اپی مثال میں ہم دو فریک نمبرز یوزر سے لے رہے ہیں اور پھر یہ کندیشن لگائی ہے کہ کیا پہلا نمبر درس سے نمبر سے تقسیم ہو جاتا ہے یا نہیں۔

Nested-if شیٹ میٹ:

آپ اپنے پروگرام میں جہاں ضرورت ہو تو شیٹ میٹ استعمال کر سکتے ہیں۔ اس طرح ایک if شیٹ میٹ میں اور بھی if شیٹ میٹ میں استعمال کی جاسکتی ہیں۔ اس کا جز لایک یہ ہے۔

```

if (condition)
{
    if(condition)
    {
        if Body;
    }
    statement;
}
else
    statement;

```

Nested if شیٹ منٹ میں سب سے پہلے باہر والی if شیٹ میٹ چک کی جاتی ہے اگر وہ درست ہو تو پھر کنٹرول اس کے اندر لکھی ہوئی کندیشن شیٹ میٹ کو دے دیا جاتا ہے اور یوں یہ پر اس چلتا ہے۔ جب اس میں لکھی ہوئی شیٹ میٹ ایک یہ کیوٹ ہو جاتی ہے تو کنٹرول پر چلی کندیشن شیٹ میٹ کو ٹرانسفر کر دیا جاتا ہے۔ آئیے اس کی ایک آسان سی مثال دیکھتے ہیں۔

مثال نمبر 2.3 شیٹ میٹ

```

void main()
{
    int var1,var2,var3,min;
    cout <<"Enter three numeric values:" ;
    cin >>var1 >>var2 >>var3;
    if(var1<var2) //var1 is less than var2
        if(var1<var3) //var1 is less than var3
            min=var1;      //min=var1
        else
            min=var3;      //min=var3
}

```

```

else
if(var2<var3) //var2 is less than var3
min=var2;      //min=var2
else
min=var3;      //min=var3
cout <<num <<"The minimum number is:" <<min;
}

```

اب فرض کریں کہ اب مندرجہ ذیل ان پڑ دیتے ہیں۔

Enter Three values in Numbers: 212, 137, 41

The minimum number is: 41

اب اس کو سمجھنے کی کوشش کریں سب سے پہلے (var1>var2) کنڈیشن ایگزیکیوٹ ہو گئی اسی میں (var1>var2) کنڈیشن سے یہ بھی غلط ہو گی۔ اب ہماری دوسری کنڈیشن else ایگزیکیوٹ ہو گا اور سب سے کم نمبر سکرین پر ڈسپلے ہو جائے گا۔ اس کے بعد اگر آپ اس طرح ولیوز ورچ کرتے ہیں۔

Enter Three values in Numbers: 41, 137, 212

The minimum number is: 41

اس کیس میں سب سے پہلے (var1>var2) کنڈیشن چیک ہو گی یہ درست ہے اس کے بعد دوسری کنڈیشن (var1>var3) ایگزیکیوٹ ہو گی وہ بھی درست ہے اس کے بعد والی سیٹ میٹ میٹ ایگزیکیوٹ ہو گی جس میں var1 var2 var3 دو یہی min دیکھ دیا گی اور یوں سب سے چھوٹا نمبر حاصل ہو جائے گا۔

سیٹ میٹ switch:

یہ سیٹ میٹ if اسیٹ میٹ کی طرح کام کرتی ہے۔ فرق صرف اتنا ہے کہ یہ ایک سیٹ میٹ میں کچھ کنڈیشن کی کمی ولیوز کو چیک کرنے سے سہولت فراہم کرتی ہے۔ بہر حال یہ if اور else if کا تبادلہ ہے۔ اس کو لکھنے کا جزیل طریقہ یہ ہے۔

Switch(expression)

```

{
Case value1;
Statement;
break;
Case value2;
break;
:
default:
}

```

```
    default statement;
```

```
}
```

سٹیٹ مینٹ میں آپ کوئی اہم باتوں کا خیال رکھنا ہوتا ہے کہ اس میں آپ کے ویری اسٹبل کی ناچ پہلی ایکسپریشن کی ولیو، short، int یا char یا byte، int میں ہوئی چاہئے۔ اور ہر case ولیو کو منفرد ہونا چاہئے۔ یہ ولیو مستقل ہوں یہ اعشاری نظام وغیرہ کو ہینڈل نہیں کرتیں۔ اس میں یہ ہوتا ہے کہ ایکسپریشن کو ہر case ولیو کے ساتھ ملایا جاتا ہے۔ جس کی ولیو برابر ہو اس کی سٹیٹ مینٹ ایگزیکیوٹ کر دی جاتی ہے اور اگر کوئی بھی case ولیو ایکسپریشن سے نہ ملے تو پھر default (ڈیفائل) ولیو پر نٹ کر دی جاتی ہے۔ اس کی مثال نیچے درج ہے۔

مثال نمبر 2.4 سٹیٹ مینٹ switch

```
viiod main( )
{
    clrscr();
    int temp=40,temp2=30;
    int sum=temp+temp2;
    switch(sum)
    {
        case 80:
            cout <<"Your Grade is: A";
            break;
        case 70:
            cout <<"Your Grade is: B";
            break;
        case 60:
            cout <<"Your Grade is: C";
            break;
        default:
            cout <<"Your are Failed:";
    }
    getch();
}
```

جب آپ اس پروگرام کو ایگزیکیوٹ کریں گے تو یہ temp اور temp2 کو جمع کرے گا اور ان کا رزلٹ sum ویری اسٹبل میں مشور کرے گا جس کو ہم نے بعد میں (switch) میں سے پاس کیا ہے اس پروگرام کی یہ آٹھ پڑت یہ ہوگی۔

آئیے switch کی ایک اور مثال دیکھتے ہیں۔

مثال نمبر 2.5

```

viod main( )
{
    clrscr( );
    int a; char b;
    cout <<"Enter a No. between 1-8";
    cin >>a;
    switch(a)
    {
        case 1:
        case 2:
        case 5:
        case 7:
            b = 'odd';
            break;
        case 2:
        case 4:
        case 6:
        case 8:
            b = 'even';
            break;
        default:
            b='sorry';
    }
    cout <<"Your No. is" <<b;
    getch( );
}

```

اس پروگرام میں ہم نے یوزر سے ان پٹ لی ہے کہ وہ اسے 9 کے درمیان کوئی بھی نمبر تحریر کرے۔ اب فرض کریں وہ اس سے ہر انمبر لکھتا ہے تو پھر ڈیفائل سیٹ میٹ پرنٹ ہو گی اور اگر وہ 1-8 تک کوئی نمبر تحریر کرے گا تو پھر switch ایکسپریشن کو کیس س کی ولیوں سے ملایا جائے گا اور اس کے مطابق رزٹ ڈسپلے ہو گا۔ اس مثال کی مدد سے آپ سمجھ سکتے ہیں کہ ایک ایکسپریشن کو کس طرح زیادہ کمپرس سے ملایا جاتا ہے۔

لا جیکل آپریٹرز:

لا جیکل آپریٹرز دو بولین (Boolean) ایک پریشزر یا آپریٹر کو ملانے کے لئے استعمال کے جاتے ہیں۔ یعنی فرض کریں آپ کہتے ہیں کہ آج آپ نے چائے پی ہے؟ اب اس کے صرف دو ہی جواب ہیں ہاں یا نہیں۔ اب آپ کہتے ہیں کہ میں کھانا کھاؤ گا۔ ان کو آپ ملانا چاہتے ہیں کہ اگر آپ نے چائے پی ہے تو میں کھانا کھاؤ گا۔ تو یہاں (اور) کے طور پر استعمال ہو رہا ہے۔ ایسے حالات میں ہم لا جیکل آپریٹر کا استعمال کرتے ہیں جب ہم دو کندہ پریشزر کو ملانا چاہتے ہیں۔ یہ آپریٹر مدرج ذیل ہیں۔

یہ اس وقت کام کرتا ہے جب دونوں کندہ پریشزر درست ہوں۔

And & &

اس میں صرف ایک کندہ پریشن کا درست ہونا ضروری ہے۔

OR ||

جب a غلط (0) ہو گا تو یہ کام کرے گا۔

Not !

اس کو ہم اس رسمی تبلیغ سے سمجھتے ہیں۔

& &		
A	B	A&&B
1	1	1
0	1	0
1	0	0
0	0	0

A	B	A B
1	1	1
1	0	1
0	1	1
0	0	0

!	
A	B
1	0
0	1

اس میں آپ دیکھ رہے ہیں کہ && میں جہاں دونوں ویلو 1 ہیں وہاں رزلٹ 1 ہے تو آپریٹر میں جہاں دونوں یا دونوں میں سے کوئی بھی ایک ویلو 1 ہے تو رزلٹ 1 اور ! آپریٹر میں جب 0 ہے تو 1 اور جب 1 ہے تو 0 رزلٹ ہے۔ آئیے اس کی C++ میں ایک مثال دیکھتے ہیں۔

مثال نمبر 2.6 لا جیکل آپریٹر کا استعمال

vioid main()

float average;

```
int Maths, Physics, English, Sum;
cout << "Enter Marks of Three Subjects:" ;
cin >> Maths >> Physics >> English;
sum = Maths+Physics+English;
if(average==100 && average>90)
    cout << "Your Grade is A+" ;
else
    if(average<90 && average>80)
        cout << "Your Grade is B+" ;
```

```

if(average<80 && average>70)
cout <<"Your Grade is C+";
else
cout <<"You are failed";
getch();
}

```

آئیے اب اس پروگرام کی ایگزکویٹیشن ترتیب کو چیک کرتے ہیں۔ سب سے پہلے یہ تم ان پڑ یوزر سے مانگتا ہے جو کہ نمبرز ہوں گے۔ اس کے بعد ان کو مجموع کرنے کے لئے sum ویری ایمبل میں شور کرتا ہے یا اس کی اوسمط معلوم کر کے فلوٹ ناپ کے ویری ایمبل average کو آسان کر دیتا ہے۔ اس پر ہم نے پھر چیک کیا ہے۔ فرض کریں آپ یہ ان پڑ دیتے ہیں۔

Enter Marks of Three Subjects; 70 83 91

Your Grade is B

اس میں وہ ان تینوں نمبروں کی اوسمط علوم کرنے کے بعد چیک کرے گا کہ اوسمط کیا ہے اس صورت میں اوسمط 80 سے زیادہ ہے۔ لہذا یہاں کنڈیشن غلط ہو گئی اب دوسرا چیک ہو گئی وہ درست ہے۔ اس کے اندر موجود نیٹ میٹ ایگزکویٹ ہو جائے گی اور اس کے بعد یہ پروگرام بند ہو جائے گا۔ اس پروگرام میں دونوں ایکسپریشنز چیک ہوں گے کران میں سے ایک بھی ایکسپریشن غلط ہو تو پھر وہ کنڈیشن غلط ہو جائے گی۔ آئیے اب ایک چھوٹا سا پروگرام لکھتے ہیں جس میں AND آپریٹر استعمال کیا گیا ہے۔

```

int a,b;
cout <<"Enter input";
cin >>a >>b;
if(a%b==0 || b%a==0)
cout <<"a and b are divisible";
}

```

اس پروگرام میں اگر دونوں میں سے ایک ایکسپریشن درست ہو گی تو آپ کی کنڈیشن درست تصور کی جائے گی اور اس کنڈیشن کی باذی بھی ایگزکویٹ ہو جائے گی۔

ویری ایمبل سکوپ:

کسی بھی ویری ایمبل یا ایڈمنیٹر کے سکوپ سے یہ مراد ہے کہ وہ پروگرام کے کسی حصے میں استعمال کیا جاسکتا ہے اور کون سا حصہ اسے استعمال نہیں کر سکتا۔ مثلاً آپ ایک ویری ایمبل اس وقت تک پروگرام میں استعمال نہیں کر سکتے جب تک کہ وہ ذیلکلر نہیں کیا جائے گا اور اس کا سکوپ وہی ہو گا جہاں یہ ذیلکلر کیا گیا ہو۔ اس کی اہمیت کا اندازہ آپ کو اگے اس کتاب میں ہو گا۔ اس وقت اس کو صرف سمجھنے کی کوشش کریں۔

```

void main()
{
int a=5;
int b=9;
}

```

```

{
    a=11; //You can use it here.

    b=12;
    c=3; //Error it is not in scope.

    int c;

    a=2;
    c=5;

}

b=3; //Ok you can use it.

c=9; //Error out of scope.
}

```

اس پروگرام میں ویری اسٹبل a اور b تمام پروگرام کہیں بھی استعمال کر سکتے ہیں جبکہ c کا سکوپ صرف اندر والی بریکلیں ہیں۔ اس کے علاوہ آپ اسے استعمال کریں گے تو یہ ایرشو کرے گا۔ آپ ایک بینہ میں ایک ہی کام کے لئے ویری اسٹبل ڈیلکلیر کر سکتے ہیں لیکن ان کا سکوپ مختلف ہوتا چاہئے۔ یہ کیمکن ہے آئیے اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 2.7 سکوپ Parallel vs Nested

```

#include<iostream.h>
#include<conio.h>
int a=5; //This is global variable.

void main( )
{
    int a=9;
    {
        int a=10;
        cout <<"Inside the main inner block a=" <<a <<endl;
    } //end of inner block
    cout <<"In main( ) a=" <<a <<endl;
    cout <<"global variable a=" <<::a <<endl;
    getch();
} //end of main( )

```

اب اس پروگرام کو ایگزیکوٹ کریں تو کچھ اس طرح کی آڈٹ پٹ ہوگی۔

Inside tha main inner block a=101

In main() a=9

اسی پروگرام میں ایک ویری ایبل a کے مختلف تین اڈجکٹس ہیں۔ پہلا جس کی ولیوو 5 ہے وہ گلوبل ویری ایبل ہے۔ اس کو آپ تمام فائل میں کہیں بھی استعمال کر سکتے ہیں کیسے؟ یا آپ فنکشنز کا اسز میں پڑھیں گے۔ اس کے بعد بھی a کی ولیوو 9 ہے وہ (main) کے بلاک میں کہیں بھی استعمال کیا جاسکتا ہے۔ جبکہ 101 ولیوو کے ساتھ اپنی شلائیز کیا ہوا ویری ایبل a صرف اسی بلاک میں استعمال کیا جاسکتا ہے۔ آپ نے جو a ویری ایبل (main) میں ڈیکلائر کیا ہے وہ پہلے گلوبل ویری ایبل کا سکوپ چھپا دیتا ہے اور اسی طرح اندر والے بلاک میں a ویری ایبل پہلے دونوں ویری ایبلوں کو چھپا دیتا ہے۔

اس طرح آپ جب بھی کسی چیز پر ہوئے ویری ایبل کو ایکسپس کرنا چاہتے ہیں تو اس کے لئے ریزرویشن آپریٹر (:) استعمال کیا جاتا ہے۔ جیسا کہ اوپر مثال نمبر 2 میں لکھا گیا ہے۔

لوپ: ایک لوپ کی مدد سے اپنے پروگرام کا ایک مخصوص حصہ بار بار (کئی دفعہ) ایگزیکیوٹ کر سکتے ہیں۔ آپ کا یہ حصہ اس وقت تک ایگزیکیوٹ ہوتا رہتا ہے جب تک کنڈیشن درست ہوئی جو نبی کنڈیشن غلط ہو جائے گی تو یہ لوپ ختم ہو جائے گا اور لوپ سے باہر انسلفر کر دیا جائے گا۔ C++ میں لوپ کی تین اقسام ہیں۔

- | | |
|----------|-------|
| For | (i) |
| While | (ii) |
| Do-while | (iii) |

for: لوپ آپ کے پروگرام کے مخصوص کوڈ کو محدود وقت کے لئے ایگزیکیوٹ کرتا ہے۔ یعنی یہ لوپ آپ اس وقت استعمال کرتے ہیں جب آپ کو معلوم ہو کہ یہ کوڈ اتنی دفعہ ایگزیکیوٹ ہونا چاہئے۔ ہر لوپ کے تین حصے ہوتے ہیں۔

- لوپ ویری ایبل اپنی شلائیز کرنا (i)
- لوپ کنڈیشن یا لوپ ختم کرنے کے لئے ولیوو دینا جسے Terminating (ii)
- ویری ایبل اپ ڈیجٹ (Increment / Decrement) کرنا (iii)

لوپ کا جزو طریقہ یہ ہے۔

```
for(initialization; condition; increment/decrement)
```

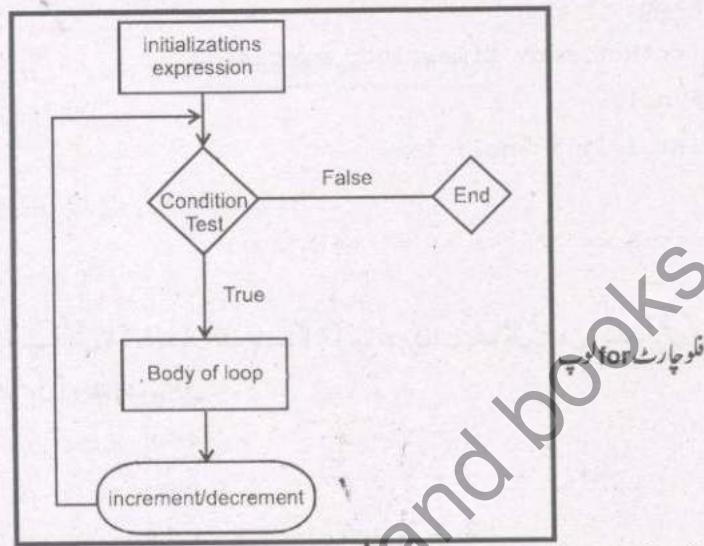
{

Body;

}

اگر آپ کے for لوپ میں صرف ایک ہی سٹیٹ میٹ ہے تو پھر { } (بریکس) کی کوئی ضرورت نہیں ہوتی۔ لوپ (for) میں تین حصے ہیں جو ایک دوسرے سے یکی کالن (:) کے ذریعے جدا کئے گئے ہیں۔ جب پہلی دفعہ لوپ شروع ہوگا تو اپنی شلائیز کنڈیشن والا حصہ ایگزیکیوٹ ہوتا ہے۔ یہ لوپ کے ویری ایبل کی ولیوو سیٹ کرنے کے لئے استعمال ہوتا ہے اور سب سے اہم بات یہ ہے کہ یہ پورشن لوپ کی زندگی میں صرف ایک بار ایگزیکیوٹ ہوتا ہے۔ اگر یہ ہر بار ایگزیکیوٹ ہو تو پھر ہر دفعہ ویری ایبل کی ولیوو ہی سیٹ ہو جائے اس کے بعد کنڈیشن چک ہوتی ہے اور یہ ہر دفعہ چیک ہو گی یہ بولین ناپ کی کنڈیشن ہوتی ہے یعنی درست یا غلط۔ اگر یہ کنڈیشن درست ہوگی تو لوپ کی باؤنڈی ایگزیکیوٹ ہو گی ورنہ لوپ ختم ہو جائے گی۔ اس کے بعد تیرا حصہ

اپ ڈیٹ کا ہوتا ہے اسے Iteration بھی کہتے ہیں۔ یہ لوپ ویری ایجیل کی ویڈیو کو اپ ڈیٹ کرتا ہے۔ for لوپ کا فلوچارٹ یہ ہوگا۔



آئیے اب اس کا ایک پروگرام لکھتے ہیں جو نمبر پر نٹ کروائے

مثال نمبر 2.8 پروگرام for

```

void main( )
{
    int no,ans;
    cout <<"Enter the Table Number";
    cin >>n;
    for(int i=1;i<=10;i++)
    {
        //start for loop
        ans i*no;
        cout <<no <<"*" <<i <<"=" <<ans <<endl;
    } //End for loop
    getch();
} //end main()
  
```

اب آپ جب اس پروگرام کو ایکسیکوٹ کریں گے تو سب سے پہلے یہ ایک ان پٹ نمبر کا آپ نیبل معلوم کرنا چاہتے ہیں۔ اس کے لئے مطلوب نمبر تحریر کریں۔ اس کا نیبل پر نٹ ہو جائے گا۔

اس کے علاوہ آپ ایک اور کام یہ بھی کر سکتے ہیں کہ یوزر کو کنٹرول دیں یعنی لوپ کتنی دفعہ چلے تو اس کے لئے دو ان پٹس میں اور کنڈیشن میں دوسری ان پٹ استعمال کریں۔ مثلاً

```
int no, ans, nol;
```

```

cout << "Enter Table Number:"
cin >> no;
cout << "How many times loop execute";
cin >> nol;
for(int i=1; i<=nol; i++)
{
    cout << no << "*" << i << "=" << (i*no);
}

```

اب اس پروگرام میں لوپ یا نیچل کا آرڈر وہاں تک جائے گا جو آپ دوسری ان پٹ میں نمبر دیں گے۔ فرض کریں آپ پہلے 2 اور بعد میں 3 ان پٹ دیتے ہیں تو اس پروگرام کا آٹھ پٹ یہ ہوگی۔

$$\begin{aligned}
 2 * 1 &= 2 \\
 2 * 2 &= 4 \\
 2 * 3 &= 6
 \end{aligned}$$

اسی طرح آئیے ایک پروگرام لکھتے ہیں جس میں بوزہ ان پٹ دیتا ہے اور ہم نے اس کا فیکٹریال (factorial) معلوم کرنا ہے۔

مثال 2.9 فیکٹریال معلوم کرنا

```

void main()
{
    int no, factor=1;
    cout << "Enter a +ve Number:";
    cin >> no;
    for(int i=2; i<=no; i++)
        factor *= i;
    cout << no << " factorial is=" << factor;
    getch();
}

```

اس پروگرام میں آپ جو ثابت نمبر تحریر کریں گے اس کا فیکٹریال معلوم کرے گا فرض کریں آپ 5 نمبر ان پٹ کے طور پر دیتے ہیں تو یہ سب سے پہلے no کو ضرب دے گا اور رزلٹ اس میں شور کر دے گا اور یہ اس وقت تک ان دونوں ویری ایبلز کو ضرب دیتا رہے گا جب تک کنڈیشن غلط نہیں ہو جائے گی۔ یعنی اس کی ولیو 5 (no==5) نہیں ہو جائے گی پھر بعد میں اس کی آٹھ پٹ یہ دے گا۔

$$5 \text{ factorial is} = 120$$

لوپ: Nested-for

دوسری پروگرام نگ لیکوچر کی طرح C++ میں Nested For Loop کی سہولت فراہم کرتی ہے۔ اس کا طریقہ بالکل as کنڈیشن کی طرح ہے۔ اس for loop کے فچر ز میں اضافہ ہوتا ہے۔ جب آپ for loop استعمال کرتے ہیں تو اس میں سب سے سلسلے باہر والی loop ایگزیکوٹ ہو گی

visit our website: www.pupapersbook.blogspot.com

اس کے بعد اس کے اندر والی لوپ چلے گی اگر باہر والی لوپ کی کنڈیشن درست ہوگی۔ اس کے بعد اگر لوپ کی کنڈیشن درست ہوگی تو اس کی بادی ایگزیکیوٹ ہوگی۔ آئیے اب اس کا ایک پروگرام لکھتے ہیں۔

مثال نمبر 2.10 Nested-for لوب

```
void main( )
{
    for(int i=1;i<=4;i++)
    {
        for(int j=1;j<=i;j++)
        {
            for(int k=1;k<=j;k++)
                cout <<"*";
            cout <<endl;
        } //end of inner loop
        cout <<endl;
    } //end of outer loop
    getch();
} //end main
```

اس پروگرام میں سب سے پہلے باہر والوں کا ایک گیریکیوٹ ہو گا اس کے بعد دوسرا لوپ چلے گا اور اس میں جو سیٹ میخت ہے وہ پرنٹ ہو گی۔ یہ لوپ چار دفعے چلے گا اس کے بعد کٹھوں دوسرے (inner) لوپ کوڑا انفر کر دیا جائے گا۔ اس طرح یہ لوپ اس وقت تک چلے گی جب تک اس کی کندیش درست رہے گی۔ پھر اس کے بعد کٹھوں پہلے (outer) لوپ کوڑا انفر کر دیا جائے گا اور اس اس وقت تک چلتا رہے گا جب تک outer (باہر والے) لوپ کی کندیش درست رہے گی۔ اس پروگرام کی آئندہ پت یہ ہوگی۔

* * * *

```

*
*
*
*
*
*
```

لوب: while

for بھی loop کی طرف ایک مخصوص کوڈ کو بار بار ایگزیکیوٹ کرنے کے لئے استعمال ہوتی ہے لیکن یہ اس وقت استعمال کی جاتی ہے while جب ہمیں یہ تنفرم ہیں ہم کا loop کی باڑی کتنی دفعہ ایگزیکیوٹ ہونی چاہئے۔ اس کا جzel طریقہ کار یہ ہے۔

```
while (condition)
```

```
{
    //body of loop;
}
```

اس میں سب سے پہلے کندھیش چیک کی جاتی ہے۔ اگر یہ درست ہوگی تو loop کی باڑی ایگزیکیوٹ ہوگی اور اس کے بعد پھر کندھیش چیک کی جائے گی۔ یہ سلسلہ اس وقت تک جاری رہتا ہے جب تک کندھیش درست رہے گی۔ جو نبی کندھیش غلط ہو جائے گی کنٹرول loop کے باہر ٹرانسفر کر دی جائے گی۔ نیچے اس کی ایک مثال ہے جس میں اٹ آرڈر میں چنپر ڈپلے کروائے گئے ہیں۔

مثال نمبر 2.11 while loop کا استعمال

```
void main( )
{
    int i=4;
    while(i>=0)           //start of loop
        cout <<i<<"\n";
        i--;                //decrement to control loop
    }                      //end of while loop
    getch();
}                      //end main()
```

اس پروگرام میں زوری اسٹبل کی ولیو 4 ہے اور loop میں ($i \geq 0$) لکھا ہے یعنی جب تک i کی ولیو 0 کے برابر یا زیادہ ہوگی loop کی باڑی ایگزیکیوٹ ہوگی اور باڑی میں i کی ولیو پر ٹک کی جا رہی ہے اور ساتھ ہی اس میں ایک کی کی بھی کی جا رہی ہے تاکہ while کی loop کی کندھیش غلط بھی ہو اور پروگرام exit ہو جائے۔ اس پروگرام کی آٹھ پٹ یہ ہوگی۔

4

3

2

1

0

```

while(i>5) {
    cout <<"Hello";
    i++;
}

```

تو یہ پروگرام کوئی آؤٹ پٹ نہیں کرے گا کیونکہ آپ کے پاس اس کی ولیوں 3 ہے۔ جب ہم کنڈیشن میں یہ چیک کر رہے ہیں کہ جب $i > 5$ (اپنے سے بڑا ہو) تب لوپ سٹیٹ میٹ چلے جو کہ غلط ہے۔ اس لئے پروگرام کوئی آؤٹ پٹ نہیں کرے گا۔

لوپ: do-while

آپ نے اور دیکھا کہ while لوپ میں سب سے پہلے کنڈیشن چیک کی جاتی ہے اور اگر پہلی دفعہ ہی کنڈیشن غلط ہو تو لوپ سٹیٹ میٹ ایگریکیوٹ نہیں ہوتی۔ لیکن بعض اوقات اسی صورت حال پیش آ جاتی ہے کہ آپ چاہتے ہیں کم از کم ایک دفعہ ہماری لوپ باڑی ایگریکیوٹ ضرور ہو خواہ کنڈیشن درست ہو یا غلط۔ تو اس کے لئے آپ do-while لوپ استعمال کر سکتے ہیں۔ اس کا جزئی طریقہ یہ ہے۔

```

do {
    loop Body;
} while(condition);

```

اس میں لوپ باڑی کم از کم ایک دفعہ اس لئے ایگریکیوٹ ہوتی ہے اسی میں پہلے لوپ کی باڑی ایگریکیوٹ ہوتی ہے اور اس کے بعد کنڈیشن چیک کی جاتی ہے اور اگر کنڈیشن درست ہو گی تو دوبارہ کنٹرول لوپ میں ٹرانسفر ہوں گے ایک دفعہ لوپ ختم ہو جائے گا۔ آئیے اس کی مدد سے ایک پروگرام لکھتے ہیں اور اس کی آفادیت سمجھتے ہیں۔

مثال نمبر 2.12: do-while

```

void main(void)
{
    clrscr();
    int temp=1,temp2=1,end;
    cout <<"How many times loop is executed:" ;
    cin >>end;
    do
    {
        temp*=temp2;
        temp2++;
    }
    while(temp2<=end);
    cout "\n Answer is:" <<temp;
    getch();
}

```

اس پروگرام میں do-while loop استعمال کیا گیا ہے۔ یہ پروگرام ایک یوزر سے ان پت لے رہا ہے اور آپ جو نمبر درج کریں گے اتنی دفعہ یہ loop ایگزیکیوٹ ہوگی اور اصل میں یہ اس نمبر کا فائل نمبر معلوم کرے گا۔ آئیے اس کی آوث پٹ دیکھتے ہیں۔

How many times loop is executed : 5

Answer is : 120

بریک سٹیٹ میٹ:

بریک سٹیٹ میٹ کی مدد سے آپ کسی بھی سٹیٹ میٹ کی ایگزیکیوشن ختم کر سکتے ہیں۔ جیسا کہ آپ نے اس سے پہلے switch سٹیٹ میٹ میں دیکھا ہے یہ سٹیٹ میٹ loop میں بھی استعمال کر سکتے ہیں اس سے loop کنٹرول باہر دوسرا سٹیٹ میٹ کو ترانسفر کر دیا جائے گا۔ اب جہاں تک اس ضرورت کا ممکن ہے یعنی یہ کہاں کہاں استعمال کی جاسکتی ہے تو وہ مختلف پرالجمن پر منحصر ہے اس کو آپ C++ میں کیسے استعمال کر سکتے ہیں۔ آئیے اس کیلئے ایک مثال سے مدد حاصل کرتے ہیں۔

مثال نمبر 2.13 break سٹیٹ میٹ کا استعمال

```
void main( )
{
    for(int i=1; i<=7; i++)
    {
        if(i==4)
            break; //terminate loop
        cout << "i=" << i << endl;
    } //end of loop
    cout << "loop is terminated";
    getch();
} //end of main
```

اس پروگرام میں loop کی ولیوں 1 سے 7 تک سٹیٹ کی ہے اور loop میں if کی کندیشن استعمال کی گئی ہے کہ جب $i=4$ ہو جائے تو loop ختم ہو جائے جب تک i کی ولیوں 4 نہیں ہوگی loop کی باڑی ایگزیکیوٹ ہوتی رہے گی۔ جو نبی $i=4$ ہو جائے گی loop سے کنٹرول باہر والی سٹیٹ میٹ کو ترانسفر کر دیا جائے گا۔ یوں اس پروگرام کی آوث پٹ یہ ہوگی۔

```
i = 1
i = 2
i = 3
Loop is terminated
```

سٹیٹ میٹ: continue

یہ سٹیٹ میٹ بریک سٹیٹ میٹ کے مقابلہ ہے۔ جب آپ اس کا استعمال کرتے ہیں تو آپ کے پروگرام کا وہ حصہ ایگزیکیوٹ ہوتا رہتا ہے۔ یعنی اگر آپ نے loop میں یہ سٹیٹ میٹ استعمال کی ہے تو یہ اسے کچھ دیر کے لئے نظر انداز کر دے گا۔ مثلاً

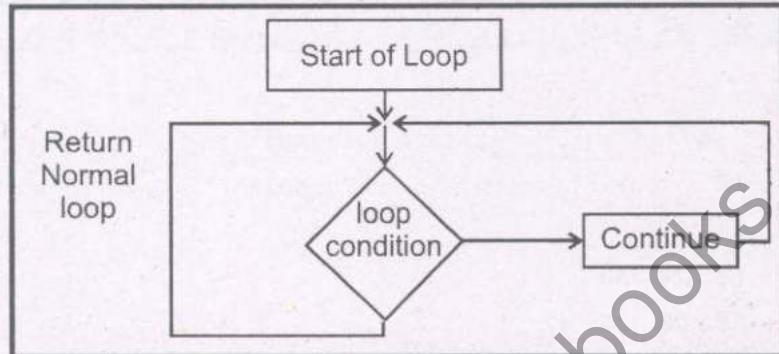
```
for(int i=1; i<=20; i++)
{
    if(i%2==0)
```

```

        continue;
        sum+=i;
    }

```

یہ پروگرام کس طرح کام کرے گا آپ continue شیٹ میٹ کے فلو چارٹ سے سمجھ سکتے ہیں۔



شیٹ میٹ: goto

آپ نے اس سے پہلے بریک، سوچ اور (continue) کا کنٹرول شیٹ میٹ کے بارے میں پڑھا ہے۔ یہ تمام شیٹ میٹ میں پروگرام کا کنٹرول واپس اسی جگہ ٹرانسفر کرتی ہیں جہاں سے یہ عام طور پر ٹرانسفر کیا جاتا ہے۔ شیٹ میٹ میں کنٹرول واپس لوپ کندیشن کو ٹرانسفر کر دیا جاتا ہے۔ جبکہ switch شیٹ میٹ میں کنٹرول درست شیٹ میٹ پر جاتا ہے اور تمام شیٹ میٹ میں کو جمپ (jump) شیٹ میٹ میں کرتے ہیں۔ اس کے علاوہ jump شیٹ میٹ کی ایک اور قسم بھی ہے جس کو goto یہی کہتے ہیں۔ اس میں آپ خود سے لیبل لگاتے ہیں کہ پروگرام کا کنٹرول کہاں ٹرانسفر کیا جائے۔ لیبل ایک ایڈیٹ نیقاٹر کی طرح ہوتا ہے جس کے بعد کا کوڈ ادا کوتا ہے یہ آپ کہیں بھی لگا سکتے ہیں۔ یہ شیٹ میٹ C++ میں کیسے استعمال کی جاتی ہے آئیے اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 2.14 Goto شیٹ میٹ

```

void main()
{
    outer:
    for(int i=1; i<10; i++)
    {
        for(int j=1; j<i; j++)
        {
            if(j>i)
                goto outer;
            cout <<(i*j) <<" ";
        } //end of inner loop
        cout <<endl;
    } //end of outer loop
    getch();
} //end of main

```

اس پروگرام میں ہم نے ایک لیبل outer کے نام سے ڈیلکلیر کیا ہے۔ اس کے بعد دو لوپس استعمال کئے ہیں۔ یعنی Nested loop میں ایک if کی کنڈیشن لگائی ہے کہ اگر (i>j) ہے تو کنٹرول یہاں سے باہر شروع میں پہلے لوپ (outer) کو رانفر کر دیا جائے۔ اگر ایسا نہیں ہوگا تو اس کنڈیشن کے بعد وابی اسیٹ میٹ ایگزیکیوٹ ہوگی اور پروگرام نارمل ایگزیکیوٹ ہوگا۔ اس پروگرام کو جب آپ ایگزیکیوٹ کریں گے تو یہ آؤٹ پٹ ڈسپلے ہوگی۔

2						
3	6					
4	8	12				
5	10	15	20			
6	12	18	24	30		
7	14	21	28	35	42	
8	16	24	32	40	48	56
9	18	27	36	45	54	63 72

(Constants and Objects):

اویجیکٹ ایک میموری کا حصہ ہوتا ہے جس کا ایک میدریک، سائز، ناپ اور ولیو ہوتی ہے۔ اویجیکٹ کا ایڈریس اصل میں میموری کی پہلی بائست کا ایڈریس ہوتا ہے۔ اویجیکٹ کے سائز سے مراد میموری میں اس کا ماحصلہ ہوتا ہے یعنی اس کے لئے کتنی بائنس ریزرو ہیں۔ ناپ میں اویجیکٹ میں سورولیو کی ناپ ہوتی ہے اور ولیو وہ کائنٹ ایٹھ ہوگا جو میموری میں محفوظ کیا جائے گا۔ اویجیکٹ کی ناپ پروگرام واضح کرتا ہے اور اس کی ولیو پروگرام خود سے کپائل نامہ پر سیٹ کرتا ہے یا رن نامہ پر آپشن یوزر کو جی دی جاسکتی ہے۔ ویری ایڈل کے بارے میں تو پڑھا ہے کہ جس کی ولیو تبدیل ہوتی رہی ہے۔ جب کہ ایسا اویجیکٹ جس کی ولیو تبدیل نہ ہو کائنٹ (مستقل) کہلاتا ہے۔ مستقل اویجیکٹ ڈیلکلیر کرنے کے لئے ساتھ یہ واضح کرنا ہوتا ہے کہ یہ ولیو مستقل ہے اور تبدیل نہیں کی جاسکتی اس کے لئے کی ورڈ استعمال ہوتا ہے۔

```
const int a=11;
```

اور ویری ایڈل کے برکس کائنٹ اس کے لئے کی ورڈ استعمال ہوتا ہے۔

```
void main()
{
    const char beep='\\b';
    const float PI=3.14;
    float n=PI/2;
}
```

فناشر:

ایسے پروگرامز جن میں حقیقی پر الہم حل کئے ہوتے ہیں اور یہ پر الہم بہت بڑے بھی ہو سکتے ہیں۔ ہم نے اب تک جو بھی پروگرام بنائے ہیں وہ صرف بنیادی پروگرام ہیں۔ اور کسی بھی اہم پر الہم کو حل کرنے کے لئے آپ کا پروگرام جتنا بڑا ہوگا اتنا ہی مشکل ہوگا اس کو سمجھنا اور اپ دیٹ یا ایڈ کرنا بھی اتنا ہی زیادہ مشکل ہوتا ہے۔ بڑے پروگرام کو بامعافی، سادہ اور آسان بنانے کے لئے پروگرامز اس کو جھپوٹے چھوٹے سے پروگراموں میں تقسیم کر visit our website: www.pupapersbook.blogspot.com

لیتے ہیں۔ یہ سب پروگرام فنکشن کہلاتے ہیں۔
 C++ میں پبلے سے بنے ہوئے فنکشن بھی ہیں۔ یہ C++ کی شینڈرڈ لامبریری میں شور ہیں۔ اس میں شرگ، میتھس (Maths) اور کریٹر سے متعلق کئی اہم فنکشن ہیں جن کو پروگرام استعمال کر سکتا ہے۔ یا اصل میں پروگرام کا کام آسان بنادیتے ہیں۔ پروگرام کسی مخصوص کوڈ کے لئے ایک فنکشن لکھتے ہیں جو ایک پروگرام میں مختلف پاؤنس پر استعمال کیا جاتا ہے۔ فنکشن کاں میں فنکشن کا نام ہوتا ہے۔ کسی بھی فنکشن کے لئے تین اہم پاؤنس اس طرح لکھتے جاتے ہیں۔

```
void func();           //function declaration/prototype
void func();           //function body/defination
{
    //body
}
fun();                //function calling
```

شینڈرڈ میتھس فنکشن:

میتھ فنکشن کی مدد سے پروگرام بنایا جی سب کیلکولیشن پر قائم کر سکتے ہیں یعنی جذر، پا اور اولاگ وغیرہ معلوم کرنا۔ یہ C++ کی شینڈرڈ لامبریری میں ڈینائیں ہوتے ہیں۔ یچہ ہم نے ایک نیجی جملہ بنایا ہے جس میں میتھس اور اہم فنکشن کی ایک فہرست ہے۔

مثال	وضاحت	(Method)
ceil(7.42) کی ولیو 1 ہے	یہ x نمبر کو اونڈا کرتا ہے یعنی اسے نزدیک ترین نمبر میں کنورٹ کرتا ہے یہ ٹرگنومیٹری میں Cosine کی ولیو معلوم کرتا ہے	ceil(x)
exp(2) 7.289 exp(2)	یہ پادر کے لئے استعمال ہوتا ہے e^x	exp(x)
fabs(-2) 2.0 fabs(-2)	یہ x نمبر کی مکمل ولیو یوریزن کرتا ہے	fabs(x)
floor(3.2) 3 میں کنورٹ کرتا ہے	یہ x نمبر کو اونڈا کرتا ہے یعنی اعشاری نظام کرتا ہے	floor(x).
fmod(4.126,2.19) اس کا بقايا یہ 1.907 ہو گا	اس میں دو ولیوز دی جاتی ہیں اور یہ x کا بقايا اعشاری نظام میں ظاہر کرتا ہے یہ مثلاً $x \% y$ ہے	fmod(xy)
log(3.276) کا جواب 0.515 ہے	اس سے x کا لاگر قسم معلوم ہو گا قدرتی میں e ہوتی ہے	log(x)
log(100.0) کا جواب 2.0 ہے	کا لاگر قسم جس کی میں 10 ہو گی	log10(x)
Pow(2,5) کا جواب 32 ہے	اس کو کہتے ہیں کہ x کی طاقت y ہے $(x)^y$	pow(x,y)
sqrt(81) کا جواب 9 ہے	اس سے مراد x نمبر کا جذر ہے	sqrt(x)
sin(1) کا مطلب 0.0174 ہے	اس سے آپ Sine کی ولیو معلوم کر سکتے ہیں	sin(x)
tan(30) کا مطلب 0.577 ہے	اس آپ ٹرگنومیٹری tan کی ولیو معلوم کر سکتے ہیں	tan(x)

اس میں آپ نے دیکھا کہ زیادہ تر میتھز میں ان کے نام کے بعد بریکیش (x) میں لکھا ہوا ہے۔ اس کا مطلب ہے آپ یہاں پر ویڈیو رج کریں جس کا آپ رزلٹ معلوم کرنا چاہتے ہیں۔ اس کو تفصیل سے آپ بعد میں درھیں گے۔ اس کے علاوہ کچھ میتھز میں دو ولیوز درج کرنے کا کہا گیا ہے۔

مثلاً (pow(x,y)) اس کا مطلب ہے کہ آپ اس میں سے دو آر گوئیں پاس کر سکتے ہیں لعنى یہاں پر آپ کو دو ڈیجیز درج کرنا ہوں گی۔ ان میختہز کا یہ فائدہ ہے کہ آپ اگر کسی نمبر کا جذر معلوم کرنا چاہتے ہیں یا لاگر قسم معلوم کرنا چاہتے ہیں تو اس سے متعلقہ فیلڈ کا کام لکھیں گے اور مطلوبہ ڈیجیز درج کریں گے ویسے اگر آپ یہ فیلڈ استعمال نہیں کرتے تو لاگر قسم معلوم کرنے کے لئے آپ کو ایک لمبا کوڈ لکھنا ہو گا اور اس کے لئے آپ کو لاگر قسم کا لئے کا اصول بھی آنا چاہتے جو کافی مشکل ہے۔ یوں یہ پروگرامز کی سہولت کے لئے ہے۔ C++ کی لائبریری math.h میں پہلے سے بنائے ہوئے میختہز ہیں۔ ان کو استعمال کرنا بہت آسان ہے اس کے لئے math.h ہیئت رفال کا کام اور ڈیجیز درج کریں۔ آئیے اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 2.15 پری ڈیفائن حابی میختہز کا استعمال

```
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main(void)
{
    double num, ans;
    cout <<"Enter number to find square root";
    cin >>num;
    ans = sqrt(num); //Calculate square root
    cout <<"The square root of " <<num << "=" <<ans;
    getch();
} //end of main
```

اس پروگرام میں سب سے پہلے دیکھیں ہم نے void main(void) کھاہے اس کا مطلب ہے کہ main میختہز تو کوئی ڈیجیز ٹرین کر رہا ہے اور نہ ہی کوئی آر گوئی پاس کر رہا ہے۔ اس کے بعد ایک یوزر سے ان پڑ لی گئی ہے جس کا جذر معلوم کرنے سے اور بعد میں sqrt(num) میں یہ ڈیجیز پاس کی ہے اور اس کا رزلٹ ans کے دیری ایبل میں شور کیا ہے۔ آئیے اس کی ایک اور مثال دیکھتے ہیں۔

اب ہم ٹرگنومیٹری پر ایلم sin2x=2sinxcosx حل کریں گے۔ اس کی ہیئت رفال آپ خود لکھیں گے ملکہ ٹرینر ہم یوں لکھیں گے۔

مثال نمبر 2.16 ٹرگنومیٹری پر ایلم sin2x=2sinxcosx حل کرنا

```
void main(void)
{
    cout <<setw(3) <<"x" <<setw(7) <<"sin2x" <<setw(10) <<"2sinxcosx";
    for (float x=0; x<2; x+=0.3)
        cout <<setw(3) <<x <<xsetw(7) <<sin(2*x)
        <<setw(10) <<(2*sin(x)*cos(x)) <<endl;
}
```

اس پروگرام میں ایک لوپ استعمال کیا ہے 2.8 سے کم رہے گا جو نبھی ڈیجیز 2 سے زائد ہو گی تو لوپ ختم ہو جائے گی۔ اس میں لوپ کے اضافے پر غور کریں ہم نے 0.3 کا اضافہ کیا ہے یعنی ضروری نہیں آپ کمی باہیشی صرف 1 کی کرس۔ اپنی ضرورت کے مطابق آپ یہ کمی باہیشی کر سکتے ہیں۔ اس

کے بعد $\sin(2x)$ کو پرنسٹ کروایا ہے جو $2\sin x \cos x$ کی ویلیو پرنسٹ کروائی گئی ہے۔ اس کی آٹھ پٹ کچھ یوں ہوگی۔

x	$\sin 2x$	$2\sin x \cos x$
0	0	0
0.3	0.564642	0.564642
0.6	0.932039	0.932039
0.9	0.973838	0.973838
1.2	0.675463	0.675463
1.5	0.141112	0.141112
1.8	-0.44252	-0.44252

یوزر ڈیفائن فنکشنز:

ہم نے پہلے بھی بتایا ہے کہ تمام فنکشنز پر وگرام کو پچھلے چھوٹے سب پر وگرام یا موز لورائز (Modularize) کرنے کے لئے استعمال ہوتے ہیں۔ ایسے تمام ویری اینڈ اینڈ فنکشنز کی ڈیفائنیشن یعنی باڑی ملکہ دیکھیر کے جاتے ہیں لوکل ویری اینڈ اینڈ فنکشن کے ہوتے ہیں لوکل ویری اینڈ اینڈ گلوبل ویری اینڈ اینڈ۔ ان کا تعارف آپ سکوپ میں پڑھ چکھی ہے۔ لوکل ویری اینڈ اینڈ صرف اسی ایریا یعنی {} میں استعمال ہو سکتے ہیں جن میں وہ ڈیکھیر کے گئے ہوں۔ فنکشن بنانے کا ایک اہم فائدہ یہ ہے کہ آپ اسے ایک پر وگرام میں کہیں بھی صرف نام لکھ کر استعمال کر سکتے ہیں۔ یعنی پورا کوڈ لکھنے کی ضرورت نہیں ہوتی صرف فنکشن کا ل کریں۔

یوزر ڈیفائن فنکشن کے دو حصے ہوتے ہیں۔ اس کا ہیڈر اور باڑی۔ ہیڈر میں فنکشن کا نام ریزن ٹائپ اور یہ ایمپر کی فہرست ہوتی ہے۔ فنکشن کی باڑی {} کے درمیان لکھی جاتی ہے اور یہ فنکشن کے ہیڈر کی پیروی کرتی ہے۔ اس میں وہ کوڈ لکھا جاتا ہے جو کوئی ایکشن پر فارم کرتا ہے اور ریزن ویلیو بھی اسی حصہ میں سیٹ کی جاتی ہے۔ آئیے یوزر ڈیفائن فنکشن کی ایک مثال دیکھتے ہیں۔

مثال نمبر 2.17 یوزر ڈیفائن سادہ فنکشن

```
#include <iostream.h>
#include <conio.h>
void func(void); //function prototype
void main(void)
{
    clrscr();
    cout <<"Main Program" <<endl;
    cout <<"Here is after function calling" <<endl;
    func(); //function calling
    getch();
}
```

```

void func(void)      //function defination
{
    float rad, result;
    const float PI=3.14;
    cout <<"Enter radius:" ;
    cin >>rad;
    result = rad*rad*PI;
    cout <<"\n Answer is:" <<result;
}

```

اس پروگرام کی افہت پخت یہ ہوگی۔

Main Program

Here is after function calling

Enter radius : 4.12

Answer is : 53.299614

اس پروگرام میں ہم نے ایک سادہ فنکشن بنایا ہے جس کا نام فنکشن نہ تو ویلیور ریٹرن کرتا ہے اور نہ ہی اس میں سے کوئی آر گومنٹ پاس کیا جا رہا ہے۔ ہم نے فنکشن (function) میں ایک کائنٹنٹ فلوٹ وری ایبل (float) کیا ہے اور دو وری ایبلز فلوٹ نائپ کے rad اور result کے ہیں۔ اس میں rad کی ویلیور سے لی جا رہی ہے جبکہ result وری ایبل میں لامچہ ریٹرن کا جواب شور کروایا گیا ہے جو بعد میں پرنسٹ کروایا گیا ہے۔

ریٹرن نائپ فنکشنز:

ایسے فنکشن جن میں سے آپ کوئی ویلیور ریٹرن کرنا چاہتے ہیں اس کی ریٹرن نائپ void کے علاوہ کچھ لکھیں گے یعنی جس فنکشن کی ریٹرن نائپ void کے علاوہ کچھ ہوگی وہ ضرور کوئی نہ کوئی ویلیور ریٹرن کرتے ہیں۔ مثلاً

```

int ab( )
{
    return axbxc;
}

```

یہ فنکشن int axbxc جو کہ ایک نائپ کی ویلیور ہوگی ریٹرن کرتا ہے۔ آپ کوئی بھی ویلیور ریٹرن کرنے کے لئے کی ورڈ return لکھتا ہوگا۔ آئیں اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 2.18 فنکشن جو ویلیور ریٹرن کرتا ہے

```

//rest of Program

double volume( );
{
    clrscr();
    char choice;

```

```

double ans;
do
{
    ans=volume();
    cout <<"volume=" <<ans <<endl;
    cout <<"Do you want to continue:";
    cin >>choice;
}
while(choice=='y'||choice=='Y');
getch();
}

double volume()
{
    double width, height, depth, result;
    cout <<"Enter width, height, depth to find volume";
    cin >>width >>height >>depth;
    result=width*height*depth;
    return width*height*depth;
}

```

آپ نے یہاں پر فنکشن کی ریٹرن ٹائپ ڈبل ظاہر کی ہے۔ اب آپ کو ڈبل ٹائپ ہی کی ویلیو ریٹرن کرنے کی اگر آپ Boolean یا char ویلیو ریٹرن کریں گے تو یہ ناممکن ہو گا۔
اس کے علاوہ جو ویری اسٹبل فنکشن کی ریٹرن کی ہوئی ویلیو کو موصول کر رہا ہے یا جس میں آپ اسے سٹور کر رہے ہیں انہیں ڈبل ٹائپ اور فنکشن کی ریٹرن ٹائپ ایک ہونی چاہئے یا پھر وہ ایک دوسرے سے compatible ہوں ورنہ رزلٹ یعنی ریٹرن کی ہوئی ویلیو میں فرق کی کریں گے۔ مثلاً آپ float ویلیو ریٹرن کرتے ہیں اور اسے سٹور int ٹائپ کے ویری اسٹبل میں کر دیتے ہیں تو اس سے ویلیو (ہو سکتا ہے اعشاریہ کے بعد والی ویلیو) میں کی یہی بھی ہو سکتی ہے۔ آپ جب اس پروگرام کو ایگزیکیوٹ کریں گے تو اس کی یہ آٹھ پڑھ ہو گی۔

```
Enter width, height, depth to find volume: 21 36 27
```

```
volume = 1512
```

```
Do you want to continue : n
```

اس پروگرام میں ہم نے ایک فنکشن (volume) بنایا ہے جس کی ریٹرن ٹائپ ڈبل ہے۔ اس میں یوزر سے تین ان پڑھ لی ہیں پھر ان کو ضرب دے کر رزلٹ کے ویری اسٹبل میں موصول کر دہ ویلیو شو کی ہے۔ بعد میں یہ رزلٹ ویلیو کو ریٹرن کیا گیا ہے۔ جسے (main) میں ڈبل ٹائپ کے ویری اسٹبل ans میں سٹور کر دیا گیا ہے۔ یہاں پر ہم نے یہ ویلیو do-while loop میں موصول کی ہے۔ یہ loop یوزر سے پوچھتا ہے کہ کیا وہ دوبارہ یہی عمل دہرانا چاہتا ہے یا نہیں؟

پیرا میٹر لسٹ:

پیرا میٹر کسی بھی فناشن کے نام کے آگے بریکش () میں لکھتے جاتے ہیں۔ ان بریکش میں ویری اسٹبلر کے نام اور ڈیٹا اسٹ پ لکھتے ہیں اور بعد میں انہیں ولیووڈی جاتی ہے۔ اس کے دوالگ الگ نام ہیں اور یہ کافی کنفیوزنگ پاؤٹ ہے۔ بعض لوگ انہیں ایک میکانالوجی تصور کرتے ہیں لیکن یہ غلط ہے۔ ان کو پیرا میٹر اور آر گومٹس کہتے ہیں۔ آئیے ان میں مبادی فرق دیکھتے ہیں۔

پیرا میٹر فناشن کی پروڈوٹ ناپ کے وقت لکھتے جاتے ہیں کہ کتنے پیرا میٹر پاس کے جائیں گے اور ان کی ناپ کیا ہوگی۔ البتہ اس کے ویری اسٹبلر کا نام بھی انہیں لکھا جاتا۔ ویری اسٹبلر کا کام اور ڈیٹا ناپ فناشن کی ڈیٹی نیشن کے وقت لکھتی جاتی ہے۔ فناشن کی باڑی میں استعمال ہوتے ہیں اور فناشن پاکی کی جانے والی ولیووکاری فرنز نہیں ہوتے ہیں۔

جبکہ آر گومٹ دیلوو کہتے ہیں یعنی جو فناشن کو پاس کی جاتی ہے۔ یہ ولیو فناشن کی ایگز کیویشن کے نام پاس کی جاتی ہے یعنی جب فناشن کا ل کیا جاتا ہے تو یہ ولیو پاس کا عامل ہے۔

C++ میں یوز رو یہاں فناشن و آر گومٹ کیسے پاس کر سکتے ہیں؟ آئیے اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 2.19 پیرا میٹر لسٹ کا استعمال

```
//Rest of Program or header files
void stdiam(double, double, double);
void main( )
{
    clrscr( );
    double width, depth, height;
    cout << "Main Program" << endl;
    cout << "Function calling with argument passing" << endl;
    stdiam(15, 7, 9);
    getch( );
}
void stdiam (double w, double d, double h)
{
    width = w;
    height = h;
    depth = d;
    cout << w << "*" << d << "*" << h << "=" << (width*height*depth);
}
```

جب آپ اس پروگرام کو ایگز کیوٹ کریں گے تو اس کا رزلٹ یہ ہو گا۔

Main Program

$$15 * 7 * 9 = 945$$

اس پروگرام میں ہم نے تین گوبل دیسی ایبلز width, height, depth ذیل ناپ کے ذیکر کے ہیں اور ایک فنکشن stdiam() نیا ہے جس میں ذیل ناپ کے تین پیروں ایمیٹر پاس کئے ہیں۔ آئیے ایک اور مثال دیکھتے ہیں۔

مثال نمبر 2.20 ریزن ناپ اور پیروں ایمیٹر

```
//header files
int volume();
void values(int,int);

main()
{
    clrscr();
    int number1;
    int number2;
    int answer;
    char ch;
    do
    {
        cout <<"Main method" <<endl;
        cout <<"Function calling" <<endl;
        values(10, 31);
        answer=volume();
        cout <<"Result is:" <<answer;
        cout <<endl; <<"Do you want to countinue?" ;
        cin >>ch;
    }
    while(ch=='y'||ch=='Y');

    cout <<"End of Program";
    getch();
}

void values(int r, int h)
{
    number1 = r;
    number2 = h;
}
```

```

int volume( )
{
    return number1*number1*number2;
}

Main method
Function calling
Result is = 3100
Do you want to continue?
n
End of Program

```

اس پروگرام کی آؤٹ پٹ یہ ہو گی۔

اس پروگرام میں ہم نے دو فنکشن بنائے ہیں ایک جو ویلور بیٹن کرتا ہے (int value) اور دوسرا جس میں سے پیرامیٹر پاس کے لئے ہیں۔ اس میں ہم نے دو فنکشن کے پیرامیٹر کے پیرامیٹر پاس کے ہیں۔ اس کے بعد ہم نے (main) پروگرام میں دونوں فنکشن do-while کے لوب میں call کئے ہیں۔ do کی لوب ہم سے یوزر کو ایک اضافی سہولت دینے کے لئے استعمال کیا ہے۔ اس کی مدد سے ہم یوزر سے یہ آپشن پوچھ رہے ہیں کہ کیا آپ یہ پروگرام دوبارہ ایگزیکیوٹ کرنا چاہتے ہیں یا نہیں۔ اگر یوزر یا ۲ لکھے گا تو آؤٹ پٹ سکرین پر دوبارہ ریسٹ آجائے گا ورنہ پروگرام بند ہو جائے گا۔

اس کے علاوہ آپ ایسا فنکشن بھی لکھ سکتے ہیں جو کوئی ویلور بیٹن کرتا ہو اور آر گومینٹ بھی پاس کرتا ہو۔ کیسے؟ آئیں اس کی ایک مثال دیجئے ہیں۔

مثال نمبر 2.21 فنکشن میں آر گومینٹ پاسنگ برے بیٹن ناچ کا استعمال

```

//header files
int factor(int);
void main(void)
{
    int no,result;
    cout <<"Enter any Positive number to find factorial:" ;
    cin >>no;
    result = factor(no);           //function calling
    cout <<"Factorial=" <<result;
    getch();
}

int factor(int fxo)
{

```

```

int a, temp=1;
for (a=f xo; a>0; a--)
temp = a*temp;
return temp;
}

```

اس پروگرام میں ہم نے ایک فنکشن (factor) کے نام سے بنا یا ہے جو ایک ویجو انٹ ٹائپ کی ریٹن کر رہا ہے اور int ٹائپ کی ویجو پاس کر رہا ہے۔ اس پروگرام میں ہم ایک نمبر یوزر سے بطور ان پٹ لے رہے ہیں اس کا ہم فیکٹریوں معلوم کر رہے ہیں۔ اس پروگرام کی آٹھ پٹ یہ ہو گی فرض کریں آپ نمبر 4 لکھتے ہیں۔

Enter any Positive number to find factorial : 4

Factorial = 24

اب تک آپ نے ایک ہی ٹائپ کی ویجو ریٹن اور پاس کی ہے۔ آپ دو الگ الگ ٹائپ کی ویجو بھی ریٹن کر سکتے ہیں۔

مثال نمبر 2.22

```

#ifndef files
char grade(int);
void main(void)
{
    clrscr();
    int marks=0;
    cout << "Enter Marks";
    cin >>Marks;
    char obt-grade;
    obt-grade = grade(marks);
    cout << "Your grade is:" << obt-grade;
    getch();
} //end of main
//Function definatoin
char grade(int l-marks)
{
    switch (l-marks)
    {
        case 90:
            return 'A';
        break;
    }
}

```

```

        case 80:
            return 'B';
            break;
        case 70:
            return 'C';
            break;
        default:
            return 'F';
    } //end of switch
} //end of function

```

پرائیم نمبر معلوم کرنا:

آئیے اب ایک ایسا پروگرام لکھتے ہیں جو کوئی نمبر معلوم کرتا ہو گا۔ پرائیم نمبر ایسا نمبر ہوتا ہے جو صرف خود پر تقسیم ہو سکتا ہے۔ مثلاً 1,3,5,7,9 اور

مثال نمبر 2.23 پرائیم نمبر معلوم کرنا

```

#include <math.h>
#include <iostream.h>
#include <canio.h>
int primeno(int);
void main( )
{
    for(int i=1; i<40; i++)
    {
        if(Primeno(i))
            cout << i << "\t";
    } //end of for loop
} //end of main
int Primeno(int i)
{
    float temp;
    temp = sqrt(i);
    if (i<2) return 0;
    if (i==2) return 1;
    if (1%2==0) return 0;
    for(int a=3; a<temp; a+=2);
}

```

```

if (i%a==0) return 0;
return 1;
}

```

اس پروگرام میں ہم نے فکشن میں آنے والے نمبر a کو سے تقسیم کیا ہے۔ موڑ (%) آپ یہ کی مدد سے ہم نے یہ if کنڈیشن (i%a==0) لگائی ہے۔ اگر ایسا ہوا تو یہ نمبر اپر ائم نمبر نہیں ہو گا۔ یہاں سے اس کی وظیفہ 0 یعنی false ہو جائے گی۔ اور اگر ایسا نہ ہوا تو پھر یہ 1 یعنی true ریٹرن کرے گا کہ یہ نمبر پر ائم ہے۔ اس پروگرام کی آؤٹ پٹ یہ ہو گی۔

2 3 5 7 11 13 17 19 23 31 37

اب ہم ایک ایسا پروگرام لیں گے جو لیپ ایئر (Leap year) کے بارے میں معلومات دے گا۔ لیپ ایئر ایسے سال کو کہتے ہیں جس میں عام سالوں سے ایک دن زیادہ ہوتا ہے۔ بعض لوگ یہ کہتے ہیں کہ جو سال چار (4) پر تقسیم ہو جائے وہ لیپ ایئر ہوتا ہے لیکن 1800، 1900 یا 1700 لیپ ایئر نہیں تھے لیکن اس میں بھی ایک منظم طور پر 2000 لیپ سال تھا۔ تو یوں centennial سال بھی لیپ ایئر ہو سکتے ہیں۔ اب اس میں یہ ہے کہ ایسا سال جو چار پر تقسیم ہو لیپ ایئر ہو گا۔ لیکن centennial سالوں کے لئے یہ اصول نہیں ہے۔ ایسا سال جو 400 پر تقسیم ہو جائے وہ لیپ سال ہو گا۔ مثلاً 2000 دغیرہ۔ تو آئیے اس کے لئے ایک پروگرام لکھتے ہیں۔

لیپ ایئر معلوم کرنا

```

//header files
int isleap(int);
void main( )
{
    clrscr( );
    int temp;
    char choice;
    do
        cout << "Enter year:" ;
        cin >>temp;
        if(isleap (temp))
            cout <<temp <<"is not a leap year \n";
        else
            cout <<temp <<"is a leap year \n";
    } while (temp>1);
    getch( );
}

int isleap(int a)

```

```

    {
        return a%4==0&&a%100!=0 || a%400==0;
    }
}

```

پیرا میسٹر بائی ریفرنس پاس کرنا:

اب تک آپ نے فنکشن میں تمام پیرا میسٹر و میلو کے ذریعے پاس کئے ہیں لیکن فنکشن میں ابھی تک و میلو پاس کی ہیں۔ اس کا یہ مطلب ہے کہ پہلے و میلو پر آپ بیش پر فارم ہو گا اور پھر فنکشن ایگزیکیوٹ ہونے سے پہلے و میلو کسی مختلف پیرا میسٹر کو آسانی کی جائے گی۔ مثلاً value(a) فنکشن کا ل کیا جاتا ہے۔ اب فرض کرو a=3 یعنی a کی و میلو 3 ہے تو فنکشن کے ایگزیکیوٹ ہونے سے پہلے یہ و میلو کسی لوکل ویری ایبل کو آسانی کرنا ہوگی۔ اس کا مطلب یہ ہے کہ ویری ایبل پر فنکشن کا کوئی اثر نہیں ہے۔ اس لئے ویری ایبل a Read-only پیرا میسٹر ہے۔

بعض صورت احوال ایسا ہی ہوتی ہیں کہ آپ کو فنکشن کے پیرا میسٹر و میلو تبدیل کرنے کی ضرورت پیش آئتی ہے تو ایسا آپ پیرا میسٹر بائی ریفرنس سے کر سکتے ہیں۔ بائی ریفرنس پیرا میسٹر کے لئے فنکشن کی روکیش میں ڈینا کے ساتھ اینڈ علامت (&) ہے ampersand بھی کہتے ہیں (لکھی جاتی ہے۔ اس کی وجہ سے لوکل ویری ایبل اصل پیرا میسٹر کا ایک ریفرنس بن جاتا ہے۔ اس سے اصل پیرا میسٹر کی بجائے Read-only کی بجائے Read-write بن جاتا ہے۔ اس کا فائدہ یہ ہے کہ فنکشن لوکل ویری ایبل میں تبدیل کیا اثر اصل پیرا میسٹر پر بھی ہو گا جو سے پاس کرتا ہے۔ آئیے اس کی ایک مثال دیکھئے ہیں۔

مثال نمبر 225 پیرا میسٹر بائی ریفرنس

```

//header files

void result(int&, int&);

void main( )
{
    clrscr( );
    int temp1=71, temp2=31;
    int temp3=80, temp2=64;
    result(temp1,temp2);
    result(temp3,temp4);
    cout <<"First time call to result:" <<temp1 <<temp2 <<endl;
    cout <<"Second time call to result:" <<temp3 <<temp4 <<endl;
    getch( );
}

void(result n1, result n2)
{
    if(n1>n2)
    {
        int temp;

```

```

temp=n1;
n1=n2;
n2=temp;
}

```

جب آپ اس پروگرام کو ایگزئکیوٹ کریں گے تو یہ رزلٹ ڈسپلے ہو گا۔

First time call to result: 31 71

Second time call to result: 64 80

اس پروگرام میں ایک فنکشن کو دو نام تکال کیا ہے۔

اوپر آپ نے ایک پروگرام لکھا ہے جس میں پیر ایمیٹر باہی ریفرنس پاس کئے گئے ہیں۔ آئیے اب ایک ایسا پروگرام لکھتے ہیں جس میں ایک پیر ایمیٹر باہی ریفرنس کا ہو گا اور دوسرا باہی ویٹیو ہو گا۔ اس سے آپ کو ان دونوں میں واضح فرق بھی معلوم ہو جائے گا۔

مثال نمبر 2.26 باہی ریفرنس اینڈ باہی ویٹیو پیر ایمیٹر

```

void differ(int x,int& y);
{
    x = 72;
    y = 42;
}
void main( )
{
    clrscr( );
    int i=10, j=39;
    cout <<"i=" <<i <<",j=" <<j <<endl;
    differ(i,j);
    cout <<"After function call" <<endl;
    cout <<"i=" <<i <<",j=" <<j <<endl;
    getch( );
}

```

اس پروگرام کی آؤٹ پٹ یہ ہے۔

i = 10, j = 39

After function call

i = 10, j = 42

اس پروگرام میں differ(i,j) کا ل کیا گیا ہے اس میں ابائی ویٹیو x کو پاس کیا جا رہا ہے اور زبائی ریفرنس y کو پاس کیا ہے۔ x ایک لوکل ویری ایبل ہے جس کو اسی ویٹیو 10 آسانی کی گئی ہے جبکہ y ویری ایبل z کا تقابل (alias) ہے جس کی ویٹیو 42 ہے۔ فنکشن 72 ویٹیو کو آسان کرتا ہے لیکن

اس کا اپ کوئی اثر نہیں ہے لیکن جب لا دیری اسے 99 ویلیو 99 کو آسان کرتا ہے تو یہ اپر اثر انداز ہوتا ہے۔ اس لئے جب آپ پروگرام ایگر میکٹ کرتے ہیں تو اس کی اصل ویلیو 10 پر نت ہوتی ہے جبکہ ز کی ویلیو فنکشن میں موجودہ کی ویلیو 42 پر نت ہوتی ہے۔ پیرا میٹر بائی ریفرنس پاس کر سکتے ہیں۔ اس کے علاوہ فنکشن کے پیرا میٹر میوری میں بہت زیادہ جگہ میغیرتے ہیں۔ تو اس کے لئے بہترین طریقہ پیرا میٹر بائی ریفرنس پاس کرنے کا ہے۔ اس کے علاوہ پیرا میٹر بائی ریفرنس پاس کرنے کا ایک فائدہ یہ ہے کہ اگر آپ پیرا میٹر کی ویلیو تبدیل نہیں کرنا چاہتے تو اسے کانٹرول ریفرنس پاس کر سکتے ہیں۔ یہ پیرا میٹر وہی کام کرے گا جو پیرا میٹر بائی ریفرنس کرے گا۔ فرق صرف اتنا ہے کہ اس میں آپ پیرا میٹر کی ویلیو تبدیل نہیں کر سکتے۔ اس کو لکھنے کا طریقہ یہ ہے۔

```
int factor(int x, int y, const int& tem0);
```

یعنی جو بھی پیروگر آپ کانٹرول بائی ریفرنس پاس کرنا چاہتے ہیں ان کے ساتھ const کی ورثہ لکھنا ہو گا۔ اور بعد میں آپ فنکشن باڈی میں اس ویلیو کو تبدیل نہیں کر سکتے۔

ریکریشن:

(Recursion)

اب تک آپ نے جتنے فنکشن پڑھ لئے ہیں اپ ان کو ایک دفعہ یا دو دفعہ (main سے کال کرتے رہے ہیں۔ اس کے علاوہ C++ میں ایک اور نیکنالوجی استعمال ہوئی ہے جسے ریکریشن کہتے ہیں۔ ایسا فنکشن جو خود کو ڈاٹریکٹ یا ان ڈائریکٹ کال کرے خواہ وہ خود کو کال کر رہا ہو گا۔ یہ کو شو فنکشن کہلاتا ہے۔ ایسا فنکشن بنیادی پر الہم کو حل کرنے کے لئے لکھا جاتا ہے اور اگر پر الہم مشکل ہو تو یہ اس کو چھوٹے ٹکڑوں میں حل کرتا ہے۔ مثلاً اگر پر الہم بنیادی ہو گی تو یہ فنکشن رزلٹ ویلیو پڑھن کر دے گا اور اگر پر الہم مشکل ہو گی تو یہ اسے دھھوں میں تقسیم کرے گا۔ ایک ایسا حصہ جسے فنکشن جانتا ہے کہ کیا کرنا ہے اور کیسے یہ حل ہو گا۔ اور دوسرا جس کے بارے میں فنکشن کو معلوم نہیں ہے۔ کہ اسے کیسے حل کرنا ہے۔ ریکریشن کی مثال اس سے سمجھ سکتے ہیں۔

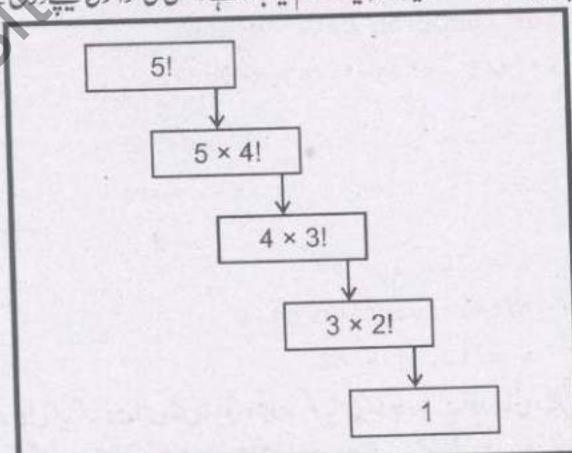
$$n! = n(n-1)!$$

یہ کسی بھی نمبر کا فیکٹوریل معلوم کرنے کا فارمولہ ہے اور اس کی مدد سے آپ کسی بھی نمبر کا فیکٹوریل معلوم کر سکتے ہیں۔ یہ اس طرح کام کرتا ہے۔

5!

$5 \times (4!)$

یہ پروگرام کس طرح کام کرتا ہے یا ایک نمبر کا فیکٹوریل نمبر کیسے معلوم کیا جاتا ہے۔ اس کی حرആ رکی یچے دیکھ جو۔



یہ اس طرح اس وقت تک کال ہوتا ہے جب تک کہ آخری ویلیو 1 نہیں ہو جاتی۔ آئیے اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 2.27 ریکرونٹش

```
#include<conio.h>
#include<iostream.h>
#include<iomanip.h>
unsigned long factor(unsigned long);
void main(void)
{
    for(int i=1; i<=10; i++)
        cout <<i <<"!=" <<setw(3) <<factor(i) <<endl;
}
unsigned long factor(unsigned long a)
{
    if(a<=0)
        return1;
    else
        return a*factor(a-1);
}
```

اس پروگرام کی آٹھ پٹ کچھ یوں ہوگی۔

1! = 1
 2! = 2
 3! = 6
 4! = 24
 5! = 120
 6! = 720
 7! = 5042
 8! = 40320
 9! = 362880
 10! = 362880

اس طرح آپ جس نمبر کا چاہیں فیکور میں معلوم کر سکتے ہیں۔ اس کے علاوہ آپ اس فنکشن کی مدد سے Fibonacci سیریز بھی معلوم کر سکتے ہیں۔

سیریز یہ ہوتی ہے۔

0, 1, 1, 2, 3, 4, 5, 8, 13, 21,

یہ سیریز صرف (0) سے شروع ہوتی ہے اور اس کے بعد ہر نیا نمبر پہلے دو نمبر کو جمع کر کے کالا جاتا ہے۔ اس کا فارمولہ یہ ہے۔

$$\text{fibonacci}(n) = \text{fibonacci}(n-1) + \text{fibonacci}(n-2)$$

آئیے اب اس کی ایک مثال لکھتے ہیں۔ جس میں یور رکوئی بھی نمبر تحریر کرے گا اور ہم اس کی fibonacci سیریز معلوم کریں گے۔

مثال نمبر 2.28 Fibonacci سیریز معلوم کرنا

```
//header files
long fiboseries(long);
void main()
{
    clrscr();
    long answer, temp;
    cout << "Enter a number";
    cin >> temp;
    answer=fiboseries(temp);
    cout << "fibonacci(" << temp << ") = "
    result << endl;
    getch();
}
long fiboseries(long a)
{
    if(a==0||a==1)
        return a;
    else
        return fiboseries(n-1)+fiboseries(n-2);
}
```

اس پروگرام کو ایز کیوٹ کریں اور رزلٹ پر غور کریں۔ فرض کریں کہ یہ در 10 نمبر لکھتا ہے۔

Enter a number: 10

fibonacci(10) = 55

اس میں یہ ہے کہ آپ کے پاس پہلے دو نمبر کے بعد یہ نمبر 55 آئے گا۔ اس طرح آپ کسی بھی نمبر کا رزلٹ معلوم کر سکتے ہیں۔

فناش اور لوڈنگ:

میں آپ ایک ہی کام کے کئی فناش بنا سکتے ہیں لیکن ان کی پیور ایمپلیکیٹ مختلف ہونی چاہئے۔ یعنی فناش کے نام ایک ہو سکتے ہیں لیکن پیور ایمپلیکیٹ مختلف ہو گی تو C++ اسے الگ الگ فناش تصور کرے گی۔ اس کو فناش اور لوڈنگ کہتے ہیں۔ جب ایسے فناش کاں کئے جاتے ہیں تو C++ کپاکٹ اس کے پیور ایمپلیکیٹ ناہیں، وہیا اور تعداد سے ان میں تمیز کرتا ہے کہ کون سا فناش کاں کیا گیا ہے۔ آئیے اس کے لئے ایک پروگرام کی مدد حاصل کرتے ہیں۔

مثال نمبر 2.29 اور لوڈ فنڈسکنٹر

```

//header files
int result(int,int);
int result(int,int,int);
void main()
{
    clrscr();
    cout <<"Sum of two numbers=" <<result (15,7);
    cout <<endl;
    cout <<"Sum of three numbers=" <<result (7,13,8);
    getch();
}

int result(int a, int b, int c)
{
    return a+b+c;
}

int result(int a, int b)
{
    result a+b;
}

```

اس پروگرام میں دو فنڈسکنٹر ایک ہی کام result سے بنائے گئے ہیں لیکن ان کے پیرا میٹر کی تعداد مختلف ہے۔ اس نے C++ کا پاکر انہیں الگ فنڈشن تصور کرے گا۔ ایک بات اور نوٹ کریں کہ ریزن ٹائپ کا اس پر کوئی اثر نہیں ہوتا۔ اب اگر آپ تمیں پیرا میٹر والے فنڈشن کا ل کرنا چاہتے ہی تو فنڈشن کا ل کے وقت تم ویلیو تحریر کریں۔ اس کے علاوہ فنڈشن کے پیرا میٹر کی تعداد بھی ایک جیسی ہو سکتی ہے لیکن اس کے لئے یخیز وری ہے کہ ان کی ڈیٹا ٹائپ پہلے فنڈشن سے مختلف ہو۔ مثلاً

```

int result (int,int);
double result(int, char);

```

اوپر والے پروگرام میں ہم نے دو پیرا میٹر والے فنڈشن کا ل کیا ہے اور بعد میں تم پیرا میٹر والے فنڈشن کا ل کیا ہے۔ یوں اوپر والے فنڈشن کی آؤٹ پٹ یہ ہو گی۔

Sum of two numbers = 22

Sum of three numbers = 28

ڈیفائلٹ آر گو منٹس:

C++ میں آپ کئی آر گو منٹس دے سکتے ہیں اس کے لئے آپ کو ڈیفائلٹ ویلیو تحریر کرنا ہوں گی۔ آپ سوچ رہے ہوں گے کہ یہ کیا ہے اور کس

طرح ہو گا؟ تو آئیے اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 2.30

```
//header files
float result(float, float=0, float=0)
void main( )
{
    float temp=1.02;
    cout << "result(temp,3) = " << result(temp,3) << endl;
    cout << "result(temp,3,5) = " << result(temp,3,5) << endl;
    cout << "result temp = " << result(temp) << endl;
    getch( );
}
float result(float a, float b, float c)
{
    return a+(b+(c*a*b))*c;
}
result(temp,3)=7.14112
result(temp,3,5)=12.241199
result(temp)=1.02
```

اس پروگرام کی آٹھ پٹ یہ ہو گی۔

اس پروگرام میں ہم نے فناشن کے دو پیر ایمیٹر کو ظیفالٹ کیا ہے جس کی وجہ سے ہم ان دو پیر ایمیٹر کی ویلوز اگر درج نہ کرنا چاہیں تو تب بھی ممکن ہے اور آپ نے دیکھا کہ ہم نے اوپر مثال میں پہلے دو ویلوز دی ہیں پھر تین اور آخر پر صرف ایک ویلوز دیکھی ہے۔

مشق

سوال نمبر 1: ایک ایسا پروگرام لکھئے جو اس طرح کی آؤٹ پٹ ڈیلے کرواتا ہو؟

(a) *
* *
* * *
* * * *
* * * * *
* * * * *

(b) * * * * *
* * * * *
* * * * *
* * * * *
* * * * *

سوال نمبر 2: ایک ایسا پروگرام لکھیں جو ایک سادہ میکرو کمپیوٹر کا کام کرے یعنی دو نریک و میکرو یوزر سے لے کر اور آپریٹر بھی یوزر سے مانگے کہ وہ ان دونوں یوزر پر کون سا آپریٹر فارم کرنا چاہتا ہے۔ یہ سوچ شیٹ میٹ کی مدد سے حل کریں۔

سوال نمبر 3: ایک ایسا پروگرام تحریر کریں جس میں یوزر سینٹی گرید درجہ خوارث کو فارم ہائیٹ یا فارم ہائیٹ کو سینٹی گرید میں تبدیل کر سکے۔

سوال نمبر 4: اس پروگرام کی آؤٹ پٹ کیا ہوگی؟

```
for(int i=1; i<5; i++)  
{  
    for(int j=0; j<i; j++) {  
        for(k=0; k<j; k++)  
            cout << "+";  
        cout << endl;  
    }  
    cout << endl;  
}
```

سوال نمبر 5: اس مساوات کو فنکشن کی مدد سے حل کریں۔

$$(n, k) = (n-1) (n-2) \dots \dots \dots (n-k+2) (n-k+1)$$

اس کی آؤٹ پٹ یوں ہوگی۔

0	0		
0	1	0	
0	1	1	0

0	1	2	2	0
0	1	3	6	0
0	1	4	12	24
			24	0

سوال نمبر 6: ایک ایسا پروگرام لکھیں جو ایک نمبر n کی پاور معلوم کرے۔ یہ نمبر n یوزر تحریر کرے اور اس کی پاور p میں آپشن ہو کہ وہ یوزر خود واضح کرنا چاہتا ہے یا نہیں۔ اگر یوزر وہ پاور خود تحریر نہ کرے تو پھر یہ پاور p کی ولیمبو 3 ہو۔

سوال نمبر 7: آپ نے اس مشہور فارمولہ کے بارے میں پڑھا ہوگا۔ اس کو حل کریں۔

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

سوال نمبر 8: مساوات کا فارمولہ ہے یہ مساوات یوں ہوتی ہے۔ $x^2 + bx + c = 0$ یوں یوزر تین نمبر تحریر کرے گا اور اس quadratic فارمولہ کی مدد سے حل کئے جانے چاہئیں۔

سوال نمبر 8: یوزر سے ایک ان پٹ میں اور معلوم کریں کہ کیا وہ پرائم نمبر ہے یا نہیں اور کیا یوزر مزید اس پروگرام کو جاری رکھنا چاہتا ہے یا نہیں؟

سوال نمبر 9: مختصر جواب دیں۔

اور `do-while` میں کیا فرق ہے؟ (i)

اور `else-if`, `if`, `switch` کس نے ابھار جوتے ہیں؟ (ii)

ریکرچ فنکشن کیا ہوتا ہے؟ (iii)

جوابات

● جواب

Ans (a)

```
#include<conio.h>
#include<iostream.h>
void main( )
{
    clrscr( );
    for(int i=0; i<6; i++)
    {
        for(int j=0; j<6; j++)
        cout <<"*";
        cout <<endl;
    }
    getch( );
}
```

Ans (b)

```
#include<conio.h>
#include<iostream.h>
void main( )
{
    clrscr( );
    for(int i=6; i>0; i--)
    {
        for(int j=6; j>0; j--)
        cout <<"*";
        cout <<endl;
    }
    getch( );
}
```

جواب : 2

```

#include<conio.h>
#include<iostream.h>
#include<iomanip.h>
void main( )
{
    clrscr( );
    int temp1,temp2;
    double result;
    char op;
    cout <<"Enter 1st number:" ;
    cin >>temp1;
    cout <<"Enter 2nd number:" ;
    cin >>temp2;
    cout <<"Enter operator(i.e +):";
    cin >>op;
    {
        case '+':
            result = temp1+temp2;
            cout <<"Answer=" <<result;
            break;
        case '-':
            result = temp1-temp2;
            cout <<"Answer=" <<result;
            break;
        case '*':
            result = temp1*temp2;
            cout <<"Answer=" <<result;
            break;
        case '/':
            result = (float)temp1/temp2;
            cout <<"Answer=" <<setprecision(2) <<result;
    }
}

```

```

        case '%':
            result = templ%temp2;
            cout <<"Answer=" <<result;
            break;
        default:
            cout <<"\n Sorry unknown operator:" ;
            break;
        }
    getch( );
}

#include<conio.h>
#include<iostream.h>
#include<iomanip.h>
void main( )
{
    clrscr( );
    double temp, result;
    int option;
    cout <<"Press 1 to convert celsius to fahrenheits:\n"
         <<"Press 2 to convert fahrenheits to celsius:" ;
    cin >>option;
    switch(option)
    {
        case 1:
            cout <<"Enter temperature in celsius:" ;
            cin >>temp;
            result = (9.5/5.0*temp)+32;
            cout <<"In fahrenheits temperature=" <<result;
            break;
        case 2:
            cout <<"Enter temperature in fahrenheits:" ;
            cin >>temp;
}

```

```

result = (temp-32)*5/9;
cout << "In celsius temperature=" << result;
break;
default;
cout << "Sorry wrong choice:" ;
break;
}
getch();
}

```

④: اس پروگرام کی آٹھ مسالوں میں سے ایک۔

```

*
*
*
*
*
*
*
*
*

```

⑤: جواب

```

#include<conio.h>
#include<iostream.h>
#include<iomanip.h>
int factorial(int a)
{
    int f=1;
    if (a < 0)
        return 0;
    while (a > 1)
        f *=a--;
    return f;
}
int permutation(int a, int b)
{

```

```

if (a < 0 || b < 0 || b > a)
    return 0;
else
    return factor(a)/factor(a-b);
}

void main(void)
{
    clrscr();
    int temp;
    cout <<"how many permutation(s) you want:\n";
    cin >>temp;
    for(int i=-1; i (temp; i++)
    {
        for(int j=-1; j<=i+1; j++)
            cout <<" " <<permutation(i,j);
        cout <<endl;
    }
    getch();
}

```

جواب :

```

#include<conio.h>
#include<iostream.h>
int prime(int);
void main()
{
    clrscr();
    int temp, power, answer, choice;
    cout <<"Enter a Number:" ;
    cin >>temp;
    cout <<"Press 1 to enter Power \n"
    <<"otherwise press any key:" ;
    cin >>power;
    answer=pow(temp,power);
}

```

```

cout << "With power (" << temp << ", " << power << ") \n Answer = " << answer;
}
else
{
    cout << "\n With default power 3:";
    answer = pow(temp, 3);
    cout << "\n power (" << temp << ", 3) \n Answer = " << answer;
}
getch();
}

```

7 جواب :

```

#include<conio.h>
#include<iostream.h>
#include<math.h>
void main(void)
{
    clrscr();
    cout << "Solving ax^2+bx+c=0" << endl;
    int a, b, c;
    double temp1, temp2, temp3;
    in:
    cout << "Enter coefficient a:" ;
    cin >>a;
    cout << "Enter coefficient b:" ;
    cin >>b;
    cout << "Enter coefficient c:" ;
    cin >>c;
    if(a==0)
    {
        cout << "This is not Quadratic Equation: Enter again";
        goto in;
    }
    cout << "the Equation is: "<<a <<"x^2+" <<b <<"x+" <<c <<"=0\n";
}
visit our website: www.papersbook.Blogspot.com

```

```

temp1 = b*b-4*a*c;
{
    cout <<"This Equation has no real solution:" ;
    getch( );
    exit(0);
}

temp2 = (-b+sqrt(temp1))/(2*a);
temp3 = (-b-sqrt(temp1))/(2*a);
cout <<"The Two possible solution are=" <<temp2 <<"," <<temp3;
getch( )
}

```

⑧: پروگرام یوزر سے ان پڑ لے گا اور پھر بتائے گا کہ کجا یہ پرائم نمبر ہے یا نہیں۔

```

#include<conio.h>
#include<iostream.h>
int prime(int);
void main()
{
    clrscr();
    int no;
    cout <<"Enter a number:" ;
    cin >>no;
    if(prime(no))
        cout <<no <<"is a prime number:" ;
    else
        cout <<no <<"is not a prime number:" ;
    getch();
}

int prime(int temp)
{
    if(temp<2) return 0;
    if(temp==2) return 1;
    if(temp%2==0) return 0;
    for(int i=3; i<temp; i+=2)

```

```

if(temp%l==0) return 0;
return 1;
}

```

9: جواب

- (i) while لوب میں پہلے کندیش چیک ہوتی ہے اور اس کے بعد لوب باڑی ایگز کیوٹ ہوتی ہے اور اگر کندیش غلط ہوگی تو لوب نہیں چلے گی۔ جبکہ do-while لوب میں کم از کم ایک دفعہ لوب ضرور ایگز کیوٹ ہوتی ہے کیونکہ اس میں لوب کندیش بعد میں چیک ہوتی ہے۔
- (ii) یہ تینوں کنٹرول سٹیٹ میں ہیں۔ # اسٹیٹ میٹ میٹ میں آپ اپنی ایکسپریشن چیک کرتے ہیں کہ کیا وہ درست ہے یا نہیں۔ اگر کندیش درست ہوگی تو اس سے مختلف سٹیٹ میٹ ایگز کیوٹ ہوگی ورنہ نہیں۔ # else کندیش # کندیش کے فچر میں اضافہ کرتی ہے اس کی مدد سے آپ ایک سے زیادہ کندیش چیک کرتے ہیں یعنی پہلی کندیش اگر غلط ہے تو دوسرا چیک ہو۔
- (iii) switch سٹیٹ میٹ کنٹرول سٹیٹ میٹ ہے جس میں آپ ایک ایکسپریشن کی کئی ویلیوز کو چیک کر سکتے ہیں۔
- (iv) ایسا فناشن جو خود کوڈ ائریکٹ یا ان فالز کیکٹ کال کرے۔ خواہ یہ فناشن خود سے خود کو کال کر رہا ہو یا کسی دوسرے فناشن سے خود کو کال کر رہا ہو رکھ رکھنے کے لیے کھلاتا ہے۔

باب نمبر 3

اریز اینڈ سٹرنگ

ایک ارے ایسے انجکش کی ترتیب کا ایک نام ہے جس کی ڈیٹا اپ ایک ہونی چاہئے یہ انجکش ارے کے اجزاء کھلاتے ہیں۔ C++ میں ارے کی بڑی اہمیت ہے اس میں کبھی کبھی تم کا ڈیٹا شور کر سکتے ہیں۔ C++ میں بھی اریز C کی طرح کام کرتی ہیں۔ اور یہ تقریباً ہر کمپیوٹر لینکوگ میں ہوتی ہیں اس کے علاوہ آپ اس کتاب میں سٹرنگ کے بارے میں بھی پڑھیں گے۔ سٹرنگ کا یہ فائدہ ہے کہ اس میں آپ کئی الفاظ (یعنی ایک جملہ) محفوظ کر سکتے ہیں۔ اس کے علاوہ آپ سٹرنگ سے متصل ہی اور فناشن بھی پڑھیں گے۔ اس باب میں آپ مندرجہ ذیل فناشن پڑھیں گے۔

اریز	
سرچ Linear	
پائیزی سرچ	
اریز کو ترتیب دینا	
ملیٹی پل سب سکرپٹس ارے	

اریز:

ایک ارے ایسے انجیکش کی ترتیب کا ایک نام ہے جس کی ڈیٹا ناپ ایک ہوئی چاہئے یہ انجیکش کے اجزاء کہلاتے ہیں اور ہر انجیکٹ کا ایک مخصوص نمبر ہوتا ہے۔ اسے ارے کا انڈیکس کہتے ہیں اور اس کا دوسرا کام سب سکرپٹ بھی ہے۔ دوسرے الفاظ میں ارے ایسی ڈیٹا ناپ کو کہتے ہیں جس میں ایک ہی نام کے ویری اینہل سٹور کے جاتے ہیں اور یہ میموری میں ترتیب سے انٹھی ایک ساتھ جگہ کھرتی ہے۔ آپ ایک ارے یوں ڈیلکٹر کر سکتے ہیں۔

int array[];

اکی int ناپ کی ارے بنائی ہے جس کا نام ارے رکھا ہے۔ اب اس میں a[1] سے مراد وہ طیو ہو گی جو میموری میں ارے کی پوزیشن پر موجود ہے ارے میموری میں صفر(0) سے شروع ہوتی ہے۔ ارے میموری میں یوں جگہ کھرتی ہے۔

int array[5]

10	12	8	7	13
0	1	2	3	4

اس سے آپ کو اندازہ ہو جائے کہ میموری میں ارے کس طرح جگہ کھرتی ہے۔ ارے میں پہلی لوکیشن صفر ہوتی ہے لیکن آپ نے 6 دیلوں کے لئے ارے ڈیلکٹر کی ہے تو میموری میں ان 6 لوکیشن 5-0 تک ہو گی۔ آخری دیلوں لوکیشن 5 میں محفوظ ہو گی۔ آجکل تقریباً تمام پروگرام زارے کا استعمال کی جو کیونکہ اس طرح آپ کا نام کم ضائع ہوتا ہے اور اس کے علاوہ میموری بھی زیادہ ضائع نہیں ہوتی۔ مثلاً آپ اگر 100 طالب علموں کا ریکارڈ رکھنا چاہتے ہیں تو کیا مر ایک الگ سے ویری اینہل ڈیلکٹر کریں گے اور الگ الگ ان پڑ دیں گے۔ اس کے لئے آپ کو اسے کا استعمال کرنا ہو گا۔ آپ اپنے پروگرام میں اسے یوں شامل کر سکتے ہیں۔

مثال نمبر 3.1 int a[5] کا استعمال

```
void main()
{
    int count[4], a=0;
    clrscr(); //to clear screen
    cout << "Enter 4 integers";
    while(a<5)
    {
        cout << "Enter digit no" <<a+1;
        cin >>count[a];
        a++;
    }
    a=0;
    do {
        cout << "\n" <<count[a];
        a++;
    } while(a<5);
}
```

```
getch();
}
```

اس پروگرام میں یوں سے ان پٹ لی جا رہی ہے اور یہ ان چار دفعہ (یعنی چار ان پٹس لی جا رہی) ہیں مگر ایک اہم بات یہ ہے کہ <> صرف ایک دفعہ کروایا ہے اس لئے کہ یہ ان پٹ ارے کی مدد سے لوپ میں لی جا رہی ہے اور بعد میں ان نمبر کے تباہ کو کاٹ دیتے کیا گیا ہے۔ اس طرح آپ ایک نام یعنی کریکٹر ارے بھی لے سکتے ہیں۔ اس کا یہ فائدہ ہے کہ آپ اس میں ایک شرٹ گک بھی لے سکتے ہیں۔ char ڈائٹا اس پ میں صرف کریکٹر کھا جا سکتا ہے اور شرٹ گک کے لئے آپ کوارے ڈیکلیر کرنا ہو گی اور شرٹ گک ان پٹ کے لئے جو مخفہ استعمال ہوتا ہے اس کا ذکر ہم پہلے ہی کرچکے ہیں۔ آئیے ایک شرٹ گک ارے دیکھتے ہیں۔

```
{
clrscr();
char name[20];
cout <<"Enter Your Name";
gets(name);
getch();
}
```

اس پروگرام میں ہم نے ایک کریکٹر ارے ڈیکلیر کی ہے جس کی لمبائی 20 ہے۔ ہم نے (clrscr) کا فنکشن استعمال کیا ہے۔ ان پٹ سکرین صاف کرتا ہے اور ان پٹ کے لئے (gets) استعمال کیا ہے۔ آئیے ارے کا ایک اسپرینٹ ہام لکھتے ہیں۔

مثال نمبر 3.2 ارے ڈیکلیر یعنی ایڈ ایڈیشن اسٹرینگ

```
void main()
{
int no[10]={0,3,5,7,9,11,13,15,17,19};
cout <<"index number" <<setw(15) <<"value" <<endl;
for(int a=0; a<10; a++)
cout <<setw(9) <<a <<setw(15) <<no[a] <<endl;
getch();
}
```

اس پروگرام میں ہم نے (setw) کا فنکشن استعمال کیا ہے اس کے لئے آپ کو ہیڈر فائل شامل کرنا ہو گی جس کا نام یہ ہے۔

#include<iomanip.h>

setw() کی مدد سے آپ دو آٹ پٹس کے درمیان فاصلہ اپنی مرضی سے سیٹ کر سکتے ہیں۔ اس پروگرام کی آٹ پٹ یہ ہو گی۔

Index no	Value
0	0
1	3
2	5

3	7
4	9
5	11
6	13
7	15
8	17
9	19

آئے اس کی ادا اور مثال دیکھتے ہیں۔

```

void main( )
{
    const int array1=7;
    int count[array1]={15,8,2,7,11,5,3};
    cout <<"array index" <<setw(12) <<"value"
    <<setw(12) <<"Magic Histogram" <<endl;
    for(int i=0; i<array1; i++)
    {
        cout <<setw(12) <<count[i] <<setw(12);
        for(int j=0; j<count[i]; j++)
            cout <<"*";
        cout <<endl;
    }
    getch();
}

```

مثال نمبر 3.3

اس پروگرام میں ایک کا نسٹنٹ ویری ایبل array1 لیا ہے۔ اس کی ڈیٹا تاپ int ہے اور اس کی ولیو 7 ہے لاکیٹ int تاپ کی ارے بنائی ہے جس کی لمبائی array1 ہے اور اسے اینی شلائیز کر دیا گیا۔ اس کے بعد for nested loop استعمال کی ہے جو array کی ولیو کے برابر پرنٹ کرواتی ہے۔ جب آپ اس پروگرام کو رن کریں گے تو اس کی آٹھ پٹ یہ ہو گی۔

array index	value	Magic Histogram
0	15	* * * * * * * * * * * *
1	8	* * * * * *
2	2	* *
3	7	* * * * *
4	11	* * * * * * * *

5

5

* * * *

6

3

* * *

یہ پروگرام اتنے پرست کرے گا جتنی اس ارے انڈکس کی ولیو ہوگی۔

مرچ: Linear

آپ نے کمپیوٹر کی تعریف میں یہ پڑھا ہو گا کہ کمپیوٹر معلومات شور کرنے کے لئے استعمال ہوتا ہے۔ خواہ ڈیا کسی بھی فارم میں ہو اور کسی بھی آرڈر میں محفوظ کیا گیا ہوشماً اریز وغیرہ۔ اس کے علاوہ آپ (پروگرام) ایسی اریز استعمال کر رہے ہیں جس میں کافی ڈیٹا شور ہے اور ایسا کثر ہوتا ہے اور ہر معلومات کو یہ پر ائمہ کی (Primary Key) ہوتی ہے جو ہر ریکارڈ کو ایک دوسرے سے مختلف کرتی ہے اور یہ کیس (Keys) ایک جیسی نہیں ہو سکتیں یعنی ان کی ولیو متفروہ ہوتی ہے۔ اس کے لئے پروگرام کو ہر ریکارڈ چیک کرنا ہوتا ہے کہ پروگرام میں کہیں کسی ریکارڈ کی تمام معلومات ایک جیسی تو نہیں ہیں کم از کم ایک فیلڈ تو منفرد ہو چکے۔ ایک ارے کے مخصوص غرض کے ذہنوں نے کے عمل کو سرچ چک کرتے ہیں۔ سرچ گ کے کئی طریقے ہو سکتے ہیں۔ مگر آپ یہاں پر دو ہم طریقے پر دھیں۔

مرچ Binary**Linear**

مرچ کا ایک آسان طریقہ یہ ہے کہ کسی بھی اوجیکٹ کو عالی لرنن کے لئے ارے کے شروع سے ہر ایٹمیٹ کو یکے بعد دیگرے (ایک کے بعد دوسرا) چیک کیا جائے اور یہ مل نہیں جاتا۔ یعنی ارے کے ہر ایٹمیٹ کو سرچ کے ساتھ ملایا جاتا ہے۔ اس کو Linear سرچ کہتے ہیں۔ آئیے C++ میں اس کا طریقہ دیکھتے ہیں۔

مثال نمبر 3.4. Linear

```
#      //libraries
void search(int& found, int& index, int arr[], int a, int ans);
main( )
{
    int count[]={12,81,2,99,5,16,18,72};
    int ans, found, location;
    do {
        cout <<"Enter number to find";
        cin >>ans;
        search(found,location,count,8,ans);           //function calling
        if(found)
        {
            location--;          // (boolean)found yes or no
            cout <<ans <<"is at index#" <<location <<endl;
        }
        else
            cout <<ans <<"is not found!";
    } while(ans !=0);
```

```

        getch( );
    }          //searching function

void search(int& found, int& index, int arr[], int a, int ans)
{
    found=index=0;
    while(!found && index<a)  {
        found=(arr[index]==ans);
    }          //end of for loop
}          //end of search

```

اس پروگرام میں جبکہ مطلوبہ نمبر لکھیں گے تو اس کا ہر دفعہ arr[index] کے ساتھ موازن کیا جائے گا۔ اگر وہ نمبر مل جائے گا تو اس کا اس سکرپٹ نمبر ڈیپلے کر دیا جائے گا لیکن کہ اس لئے کس کا نمبر یہ ہے ورنہ یہ ڈیپلے ہو جائے گا کہ نمبر نہیں ملا۔ جب آپ اس پروگرام کو ایگزیکیوٹ کریں گے تو یہ اس طرح آؤٹ پٹ ڈیپلے کرے گا۔

```

Enter number to find  99
99 is at index # 3
Enter number to find  8
8 is not found!
Enter number to find  0
0 is not found!

```

اور اس کے ساتھ ہی پروگرام ختم ہو جائے گا۔ ہم نے لوپ میں (ans!=0) کا لمحہ ہے یعنی اس وقت تک یہ لوپ ایگزیکیوٹ ہوتا رہے جب تک یور صفر (0) ان پٹ نہیں دیتا۔

Binary Search:

یہ بہت تیز سرچنگ میکنا لوگی ہے۔ یہ ارے کو دو حصوں میں تقسیم کرتی ہے اور پھر ان دو حصوں میں سے مطلوب نمبر تلاش کرتی ہے۔ میکنا لوگی کس طرح کام کرتی ہے آئیے اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 3.5. بائنری سرچ میکنا لوگی

```

#      //header files

void bsearch(int& found, int& index, int arr[], int a, int ans);
void main( )
{
    clrscr( );
    int count[]={12,10,81,9,29,71,20,0};
    int temp, answer, location;
    do {

```

```

cout << "Enter a number to find";
bsearch(answer, location, count, 8, temp);
if(answer)           //number found or not
    cout <<temp <<"is found at index#" <<location <<endl;
else
    cout <<temp <<"is not found";
while(temp !=0);
//Binary searching function
void bsearch(int& found, int& index, int arr[], int a, int ans)
{
    int temp1=0, temp2=a-1;
    found=0;
    while(!found& temp1<=temp2)
    {
        index=(temp1+temp2)/2; //enteral point
        found=(arr[index]==ans);
        if(arr[index<ans])
            temp=index+1;
        else
            temp2=index;
    }
}

```

اریز کو ترتیب دینا:

C++ لیگوچ میں اسے Sorting کہتے ہیں۔ اس کا مطلب ہوتا ہے کہ اریز ڈیٹا کو ایک ترتیب سے ظاہر کرنا اور یہ بہت سامنہ ہے۔ مثلاً سکول میں تمام طالبموں کو ان کے روپ نمبر کی ترتیب سے لکھا جاتا ہے، بلکہ اپنے تمام ہمکیس کو کاؤنٹ نمبر کے مطابق ترتیب دھاتا ہے غرض کہ تمام کپسیاں اپنے ڈیٹا کو ایک خاص ترتیب سے لکھتی ہیں۔

C++ میں آپ ڈیٹا کو ترتیب کیسے دے سکتے ہیں؟ آئیے اس کے لئے ایک پروگرام لکھتے ہیں۔

Sorting Array 3.6

```

#include <iostream.h>
void main( )
{
    clrscr( );

```

```

int count1, count2, temp;
const int size=8;
int arr[size]={1,9,3,7,11,5,0,13};
cout <<"Here is original array" <<endl;
for(count1=0; count1<size; count1++)
cout <<arr[count1];
for(count1=0; count1<size-1; count1++)
{
for(count2=0; count2<size-1; count2++)
{
if(arr[count2]>arr[count2+1])
{
temp=arr[count2];
arr[count2]=arr[count2+1];
arr[count2+1]=temp;
} //end if
} //end for
} //end for
cout <<"Here is sorting Data" <<endl;
for(count1=0; count1<size; count1++)
cout <<arr[count1];
getch();
}

```

اس پروگرام میں سب سے پہلے a[0] کا a[1] سے موازنہ کیا جائے گا پھر a[1] کا a[2] سے اس کے بعد a[3] کا a[2] کا a[1] سے اس کے بعد a[4] کا a[3] کا a[2] کا a[1] سے۔ اس طرح یہ تک جائے گا اور ہر دفعہ یہ نمبر کو ترتیب سے دے گا۔ مثلاً اگر a[1] کا a[2] سے بڑا ہو گا تو یہ اس کو a[0] پر سورکرے گا۔ اس کے لئے ہم نے for loop استعمال کی ہے اور اس میں ان نمبرز کو swap کروایا ہے۔

```

temp=arr[count2];
arr[count2]=arr[count2+1];
arr[count2+1]=temp;

```

یعنی arr[0] والی دلیل میں سورہ ہو جائے گا اس کے بعد arr[1] میں 1 جمع کیا ہے۔ یوں arr[1] میں موجود دلیل arr[0] میں آجائے گی اور arr[0] والی دلیل temp میں تھی وہ arr[1] میں آجائے گی۔ یوں یہ پروگرام کام کرے گا جب آپ اپرواہی مثال 3.6 کو اگر کیوں کریں گے اس کا رزلٹ یہ ہو گا۔

1 9 3 7 11 5 0 13

Here is original array

0 , 1 , 3 , 5 , 7 , 9 , 11 , 13

یہ سینکا لو جی یا طریقہ ہم نے مثال نمبر 3.6 میں استعمال کیا ہے۔ یہ sort bubble sort یا sink sort کہلاتا ہے۔ اب ہم ایک ایسا پروگرام بنائیں گے جو اسے میں موجود نمبرز کو کاؤنٹ کریں گے۔ یعنی کونا نمبر کتنی دفعہ اسے میں لکھا گیا ہے۔ یہ صرف آپ کی پریکش کے لئے ہے تاکہ آپ کو اریز پر زیادہ سے زیادہ مکانڈ حاصل ہو۔

مثال نمبر 3.7 اریز دیجیٹ کاؤنٹ کرنا

```
//#Header files
void main( )
{
    clrscr();
    const int size=34, countsize=10;
    int array[size]={0,1,3,2,9,5,7,9,8,4,5,6,7,8,9,2,3,4,5,6,7,
                     8,9,10,8,3,2,1,4,5,3,2,10};
    int count[countsize]={0};
    for(int i=0; i<size; i++)
        ++count[array[i]];
    cout <<"Elements" <<setw(13) <<"Repitition" <<endl;
    for(int i=1; i<contsize; i++)
        cout <<i <<setw(13) <<count[i] <<endl;
    getch();
}
```

اس پروگرام کی آؤٹ پٹ یہ ہوگی۔

Elements	Repitition
0	2
1	3
2	5
3	5
4	3
5	4
6	2
7	3
8	3
9	3

10

1

مئی میں سب سکرپٹس ارے:

آپ نے میکس میں ملٹی پل ارینز کے بارے میں پڑھا ہوگا۔ یہ ڈینا کوئی نیل میں قطاروں اور کامز کی شکل میں ڈپلے کروانے کے لئے استعمال ہوتی ہے۔ C++ میں بھی یہ اسی کام کے لئے استعمال کی جاتی ہے۔ اس کے لئے ہم دو سکرپٹس واضح کرتے ہیں۔ پہلا قطار کو ظاہر کرنا ہے جبکہ دوسرا کام کے لئے استعمال ہوتا ہے۔ C++ میں ملٹی پل سکرپٹس ارے کیسے بناتے ہیں مثلاً

```
int array[2][3]={{2,4,8},{1,3,5}};
```

اس میں بوقظہ اس اور تین کالم ہیں۔ اب اس کو پوس ڈسلے کروا پا جائے گا۔

```
for(int i=0; i<2; i++)
{
    for(int j=0; j<3; j++)
        cout <<array[i][j];
}
```

اس میں سلالوں قطاروں کو نکروں کرتا ہے جبکہ کلمہ کے لئے دوسرا لوپ لکھا گیا۔ اس کی آٹوٹ پٹ بیوں ہو گی۔

2 4 8
1 3 5

ابھی آپ نے دوستی ارے کے بارے میں پڑھا ہے لیکن اور حال ہیں تم نے صرف ارے اینی ہلا یز کروائی ہے اور بعد میں ڈپلے کروائی ہے۔ آپ اریز کے ساتھ عمل بھی کر سکتے ہیں۔ آئیے اس کے لئے الگی مثال 3.7 دیکھتے ہیں۔ جس میں دو میرکس کو جمع کر کے تیری میرکس میں محفوظ کروایا گیا ہے۔

مثال نمبر 3.8 نئن سمتی اے

```
//header files
void main(void)
{
    clrscr();
    int array[3][3],array2[3][3], array3 [3][3];
    int a, a1;
    cout <<"Enteries for 1st Matrix" <<endl;
    for(a=0; a<3; a++)
        for(a1=0; a1<3; a1++)
    {
        cout [ <<" " <<a <<" ] " <<" [ " <<a1 <<" ] =";
        cin >>array1[a][a1];
    }
}
```

```

cout << "Second Matrix" << endl;
for(a=0; a<3; a++)
for(a1=0; a1<3; a1++)
{
    cout << "[" <<a <<"]" << "[" <<a1 <<"]" =";
    cin >>array2[a][a1];
}
cout << "Resulting Matrix:" << endl;
for(a=0; a<3; a++)
for(a1=0; a1<3; a1++)
{
    array3[a][a1]=array1[a][a1]+array2[a][a1];
    cout << "[" <<a <<"]" << "[" <<a1 <<"]" = " <<array2[a][a1] <endl;
}
getch();
}

```

اب جب آپ اس پروگرام کو کپائل کرنے کے بعد ایگزئکوٹ کریں گے تو سطح میں پہنچان پہنچ لے گا پھر دو اریز کو جمع کرے گا اور جواب میں سورکرے گا۔ اس پروگرام کی آٹھ پڑی ہو گی۔

Enteries for 1st Matrix:

```

[0] [0] = 1
[0] [1] = 2
[0] [2] = 3
[1] [0] = 1
[1] [1] = 2
[1] [2] = 3
[2] [0] = 4
[2] [1] = 5
[2] [2] = 6

```

Enteries for Second Matrix:

```

[0] [0] = 7
[0] [1] = 8
[0] [2] = 9
[1] [0] = 45

```

```
[1] [1] = 46
[1] [2] = 25
[2] [0] = 12
[2] [1] = 21
[2] [2] = 0
```

Resulting Matrix:

```
[0] [0] = 8
[0] [1] = 40
[0] [2] = 12
[1] [0] = 46
[1] [1] = 48
[1] [2] = 28
[2] [0] = 16
[2] [1] = 26
[2] [2] = 6
```

اس مثال میں ہم نے ہر کالم کی ان پٹ الگ سے لی چکیا تھا یہ بھی بتایا ہے کہ آپ کون سے نمبر کے لئے ان پٹوں رہے ہیں تاکہ آپ کو پڑھ جل سکے کہ سوراہیل اریز میں کس طرح ان پٹ لی جائی ہے اور کیسے اریز سے سور کرتی ہے۔ آپ اس کے علاوہ نارمل ان پٹ بھی سکتے ہیں مگر وہ تھوڑا سا مشکل ہوتا ہے کہ ان پٹ کیسے دینی ہے۔ آئیے اب ایسے اور مثال دیکھتے ہیں۔ جس میں آپ اپنے فتنشن کو پاس کرنے کا طریقہ پڑھیں گے یعنی اسے کس طرح بطور فتنشن پیرا میٹر استعمال کی جا سکتی ہے۔

مثال نمبر 3.9 ارے کا استعمال

```
//header files
const int employee=3           //global variable
const int sale=4;               //global variable
int control=0;                 //function to get average
float average(int temp[], int count)
{
    int result=0;
    for(int i=0; i<count; i++)
        result+=temp[i];
    cout << "\n Total of Row:" << control << "=" << result << endl;
    control+=1;                  //increment 1 and again assign to control
    return (float) result/count; //return average
}
```

```

void output(int sales[][sale], int dept, int test)
{
    cout << endl << setw(22) << "[0] [1] [2] [3]";
    for(int i=0; i<dept; i++)
    {
        cout << "\n[" << i << "]" << setw(7);
        for(int j=0; j<test; j++)
            cout << setiosflags(ios::left) << setw(6) << sales[i][j];
    }           //end for loop
}           //end function

void main(void)
{
    clrscr();
    int array[employee][sales];
    cout << "Enter values as 4 number in one Row" << endl;
    cout << "And Then press enter" << endl;
    for(int i=0; i<employee; i++)
        for(int j=0; j<sale; j++)
            cin >> array[i][j];           //for input
    cout << endl;
    output(array, employee, sale);      //function call
    for(int i=0; i<employee; i++)
        cout << "\n Average of" << i << "=" << setprecision(2)
                           << average(array[i], sale);
    cout << endl;
    getch();
}

```

جب آپ اس پر گرام کو ایگریکوت کریں گے تو پر زک ڈالے ہوگا۔

Enter values as 4 number in one Row

And Then press enter

24 12 3 6

4 51 7 8

61 31 33 25

```

[0]   [1]   [2]   [3]
[0] 24    12    3    6
[1] 4     51    7    8
[2] 61    31    33   25

```

Total of Row 0 = 45

Average of 0 = 11.25

Total of Row 1 = 70

Average of 1 = 17.5

Total of Row 2 = 150

Average of 2 = 37.5

اس پروگرام میں فنکشن کو ارجمند پاس کی گئی ہے۔ فنکشن (Output) آپ کے تحریر کردہ ذیلیا کی دوبارہ آٹھ پٹ شو کرتا ہے کہ آپ نے کس سب سکرپٹ پر کوشش ویڈیو دی ہے۔ اس کے علاوہ اکامہ اور فنکشن (Average) کا ہے جو اس ارے کی اوسط ریزن کرتا ہے۔ اور اس کے علاوہ یہ ہر قطار کا رزلٹ (جس) بھی شو کرواتا ہے۔ اس فنکشن میں ایک (cout) یہ ہے۔

control+=1;

یہ ہر لائن نمبر الگ سے پرنسٹ کروانے کے لئے استعمال کیا جائے۔ اس کے علاوہ (main) فنکشن میں ہم نے ایک فنکشن (setprecision(2)) کیا ہے جو اوسط کی ریزن کردہ ویڈیو میں اعشاریہ کے بعد وائی ویڈیو کو کروال کرتا ہے اور صرف دونہ بڑی 11.25 ریزن کرتا ہے۔

(Strings):

سٹرنگ میوری میں با معانی کریکٹر زکی ترتیب کو کہتے ہیں جس کا اختتام Null ('0') کیٹر سے ہوتا ہے یا سٹرنگ کریکٹر کی ایک ارے کو کہتے ہیں اور اس کے آخر پر ('0') کریکٹر ہوتا ہے۔ اس کا مطلب یہ ہے کہ سٹرنگ کی لمبائی ہمیشہ اسی محدود کریکٹر سے ایک زیادہ ہو گی کیونکہ اس میں ایک Null کریکٹر لازمی آنا ہوتا ہے۔ آپ سٹرنگ یوں ڈیلٹیر اور اینی ہلا یہیز کر سکتے ہیں۔

```

char str[]="Sikandar";
cout <<str;

```

آئیے اس کی ایک مثال لکھتے ہیں۔

مثال نمبر 3.10 سٹرنگ

```

void main(void)
{
    char str[]="jutt";
    for(int i=0; i<5; i++)
        cout <<"str[" <<i <<"]" <<str[i] <<endl;
    getch();
}

```

```

str [0] = j
str [1] = u
str [2] = t
str [3] = t
str [4] = ' '

```

سٹرنگ لابریری:

اس کے علاوہ C++ سٹرنگ کا کچھ بیلڈ سے بننے ہوئے فنکشن کی سہولت بھی دیتی ہے۔ جس میں سٹرنگ ڈیٹا کا پی کرنا، سٹرنگ کی لمبائی معلوم کرنا وغیرہ شامل ہیں۔ اگر آپ سٹرنگ کا کوئی بھی فنکشن استعمال کرنا چاہتے ہیں تو اس کے لئے آپ کو C++ کی شینڈر ڈالابریری <string.h> استعمال کرنا ہو گی۔ نیچے نیبل میں چند اہم اور مفید سٹرنگ فنکشن کا استرتیجی دی گئی اور ان کا استعمال آپ بعد میں دیکھیں گے۔

فونکشن	وضاحت
strcmp()	یہ اس بات کی نتیجی کرے گا کہ ویری ایبل c کی ویلوسٹرنگ میں کس پوزیشن پر ہے اور اگر c کی ویلوسٹرنگ میں شامل نہیں ہو گی تو یہ Null ریٹن کرے گا۔
strcpy()	یہ سٹرنگ کی لمبائی معلوم کرنے کے لئے استعمال کیا جاتا ہے۔
strlen()	یہ سٹرنگ کی لمبائی معلوم کرنے کے لئے استعمال کیا جاتا ہے۔
strncpy()	اس میں یہ بات واضح کر سکتے ہیں کہ s2 سٹرنگ کا تناصر s1 میں کاپی ہو۔
strncat()	اس میں یہ بات واضح کر فی کے کہ s2 کا کتنا حصہ s1 سے بتائے گے کہ یہ کیٹریز کا موازنہ کرتا ہے۔
strncmp()	یہ سٹرنگ سے پہلے s1 سے strncmp(s1,s2,3) کی ویلوسٹرنگ میں پہلی موجودی کو ظاہر کرتا ہے۔
strpbrk()	یہ سٹرنگ 1 اور سٹرنگ 2 کی ویلوسٹرنگ میں پہلی مطابقتی کو ظاہر کرتا ہے اور سب سے پہلے جس انداز میں اس پر ان کی ویلوسٹرنگ ہو وہ اندیکس نمبر ریٹن کرتا ہے۔
strspn()	اس سے آپ یہ واضح کرتے ہیں کہ s2 کا کون سا کریکٹر یا ورڈ s1 کا حصہ ہے۔
strstr()	یہ دو سٹرنگ کا موازنہ کرتا ہے لیکن یہ اس بات کا لحاظ نہیں رکھتا کہ دونوں سٹرنگ کا کیس کیا ہے یعنی یہ نہیں ہے۔
strcmpi()	case sensitive

آپ نے اوپر نیبل میں C++ کی <string.h> ہیڈر فائل کے چند اہم فنکشن کے بارے میں مختصر آپڑھا۔ اب آگے آپ ان کا استعمال پڑھیں گے کہ یہ کس طرح کام کرتے ہیں۔

آئیے اب ایک پروگرام لکھتے ہیں جو یوزر سے دو ان پیس لے گا اور ان کا آپس میں موازنہ کرے گا کہ کون سا سٹرنگ بڑا ہے یا چھوٹا ہے۔

مثال نمبر 3.11 کمپریشن اسٹرنگ

```
//get( ) header file
```

```

#include<iostream.h>
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main(void)
{
    clrscr();
    char a[10], a1[15];
    int x;
    cout <<"Enter First string:";
    gets(a);           //for string input with space
    cout <<"Enter Second string:";
    gets(a1);
    x=strcmpi(a,a1);   //compare two strings
                        //without case sensitive
    if(x==0)
        cout <<a <<": is equal to: " <<a1;
    else
        if(x>0)
            cout <<a <<": is greater than: " <<a1;
        else
            if(x<0)
                cout <<"Something is wrong Try again:";
            getch();
}

```

جب آپ اس پروگرام کو کپائل کرنے کے بعد ایگزیکوٹ کریں گے تو یہ آٹھ پڑھ حاصل ہو گی۔ یہ ان پڑھ ہم نے فرضی دی ہے آپ کچھ بھی ان پڑھ دے سکتے ہیں۔

Enter First string: Welcome to Gojra

Enter Second string: I Live in Gojra

Welcome to Gojra: is greater than: I live in Gojra

ہم نے اس پروگرام میں دو کریکٹر ایز [10] اور [15] a[] اور a1[] کی چیز اور یوزر سے ان پڑھ (gets()) سے لے لی ہے اس لئے کہ cin یوزر سے پس نہیں لیتا۔ پھر ہم نے ان دونوں سترنگز کا (strcmpi(a,a1)) کی مدد سے موازنہ کروایا ہے اور اس کی ویلیو اور یہ ایسٹل میں مشورہ کی ہے کہ یہ دونوں برابر ہیں یا نہیں۔ ہم نے ((strcmpi(a,a1)) کا تکالیف نہیں کرتا کہ دونوں سترنگز کا حرف تھیں لکھنے کا ناکام بھروسہ رکھتا ہے اس کے حساب سے یعنی سوہنگہ کی

موارنے کے بغیر کرتا ہے۔ یعنی اگر آپ یہ ان پڑ دیتے ہیں۔ Case Sensitivity

```
Enter First string: HELLO
```

```
Enter Second string: Hello
```

تو یہ دونوں کو equal شو کرے گا اس لئے کہ یہ case sensitive نہیں ہے لیکن اگر آپ چاہتے ہیں کہ ہمارا فناشن پر (strcmp) فناشن استعمال کریں۔ باقی سارا عمل وہی ہے صرف یہ لائن تبدیل کریں۔

```
strcmp(a,al)
```

سٹرنگ جمع کرنا:

آئیے اب سٹرنگ کے لئے ایسا پروگرام لکھتے ہیں جو دو سٹرنگز کو (جمع) کرے گا۔ ایک سٹرنگ کو دوسرے کے آخر پر لگائے (چپا کرے) گا اور ساتھ میں ان کی لمبائی بھی شو کروائے گا۔

مثال: 3.12 strlen / Strcat فناشن کا استعمال

```
//header files
void main(void)
{
    clrscr( );
    char temp1[]="Shoaib and";
    char temp2[]="Awais";
    cout<<"Before strcat function";
    cout <<"\n First string=" <<temp1 <<", [length] =" <<strlen(temp1);
    cout <<"\n Second string=" <<temp2 <<", [length] =" <<strlen(temp2);
    strcat(temp1,temp2);
    cout <<"\n after strcat function";
    cout <<"\n 1st string=" <<temp1 <<", [length] =" <<strlen(temp1);
    cout <<"\n 2nd string=" <<temp2;
    cout <<"\n Enter a string:";
    gets(temp1);
    cout <<"temp1 <<: is length=" <<strlen(temp1);
    getch( );
}
```

اس پروگرام میں ہم نے دو کریکٹر اریز [temp1] اور [temp2] بنائی ہیں اور ان کی لمبائی واضح نہیں کی یعنی یہ ہم رن نائم پر جتنی مرضی بھی کر سکتے ہیں اور ان کا کٹھا کیا ہے اور ان کی لمبائی معلوم کی ہے اور بعد میں [temp1] میں یوزر سے ان پڑ لی اور اس سٹرنگ کی لمبائی معلوم کی ہے۔ اس پروگرام کی آٹھ پٹ یہ ہوگی۔

Before strcat function

```

First string = Shoaib and      , [length]=10
second string = Awais          , [length]=6
After strcat function
1st string = Shoaib and Awais , [length]=16
2nd string = Awais
Enter a string Yasir and Sikandar
Sikandar and Yasir           , [length]=18

```

سٹرنگ پر عمل

آپ ایک سٹرنگ پر کاپی بھی کر سکتے ہیں۔ اس کے لئے C++ کی شینڈر ڈل ابیری استعمال کی جاتی ہے۔ آئیے اس کی ایک مثال دیکھتے ہیں۔ اس میں C++ کا شینڈر فنشن (strcpy) استعمال کریں گے یہ پہلے سٹرنگ پر دوسرے سٹرنگ کو کاپی کرنے کے لئے استعمال کیا جاتا ہے۔

مثال نمبر 3.13 سٹرنگ کاپی کرنا

```

//header files
void main( )
{
    clrscr( );
    char string1[30], string2[25];
    char choice;
    do
    {
        cout <<"Enter First String:" ;
        gets(string1);
        cout <<"\nEnter Second String:" ;
        gets(string2);
        cout <<"\n Before strcpy function:" ;
        cout <<"\n 1st string is:" <<string1;
        cout <<"\n 2nd string is:" <<string2;
        strcpy(string1, string2);
        cout <<"\n After strcpy function:" ;
        cout <<"\n 1st string:" <<string1;
        cout <<"\n 2nd string:" <<string2;
        cout <<"\n Do You want to continue?" ;

```

```

    cin >>choice;
}

while(choice=='y' | choice=='Y');
}
//end main

```

اس پروگرام کی آؤٹ پٹ کچھ یوں ہوگی۔

Enter 1st string: Welcome to My book

Enter 2nd string: Ecommerce

Before strcpy function

1st string: Ecommerce

2nd string: Ecommerce

Do You want to countinue? n

آپ نے دیکھا کہ اس میں دوسرا اسٹرینگ پہلے سب پر کاپی کر دیا جاتا ہے اور پہلا اسٹرینگ ختم ہو جاتا ہے۔ اس کے علاوہ ایک اور فناش ہے یہ سٹرینگ 2 کا کچھ مخصوص ذیلا اسٹرینگ کے مخوبیت سے پر کاپی کر دے گا اس کے لئے باقی پروگرام وہی رہے گا صرف یہ لائن تبدیل کریں۔

```
strncpy(string1, string2, 6);
```

اب فرض کریں آپ اسے ایکریکووٹ کرتے ہیں تو یہ ان پڑھ مارے جائے اسی یہاں پڑھ دیتے ہیں۔

Enter 1st string: Hello How Are You

Enter 2nd string: Thanks Fine

Before strcpy function

1st string: Thanks How Are you

2nd string: Thanks Fine

Do You want to countinue? n

آؤٹ پٹ پر غور کریں کہ اس نے سٹرینگ 1 کے پہلے 6 کریکٹریز پر سٹرینگ 2 کے چھ کریکٹریز چھپاں کر دے گیں۔ سٹرینگ 2 میں کوئی تبدیلی نہیں آئی وہ بالکل ویسا ہی ہے۔

سٹرینگ کو اکھا کرنا:

آپ نے ایک سٹرینگ کو دوسرے سٹرینگ کے ساتھ ملانے کا طریقہ پہلے بھی دیکھا ہے لیکن اب ہم آپ کو اس سے تھوڑا سا مختلف طریقہ تائیں گے۔ اس کے علاوہ آپ اس پروگرام میں (strchr() فناش کا استعمال بھی دیکھیں گے۔ تو اس کے لئے ایک پروگرام لکھتے ہیں۔

مثال نمبر 3.14 strchr() اور strncat() فناش

```

//header files
void main( )
{
    clrscr();
    char string1[]="Yasir and";

```

```

char string2[]="Sikandar are best Friends";
int answer;
cout <<"Before strncat Function." ;
cout <<"\n string1=" <<string1;
cout <<"\n string2=" <<string2;
strncat(string1, string2, 8);
cout <<"After strncat Function." ;
cout <<"\n string1=" <<string1 <<", [length] = " <<strlen(string1);
cout <<"\n string2=" <<string2 <<", [length] = " <<strlen(string2);
cout <<endl;
cout <<"\n Use of strchr end strstr";
answer=(int) strchr(string1, 'd');
cout <<"\n strchr(string1, 'd') is at location=" <<ans-(int)string1;
cout <<"\n Enter string to find from string2:" ;
gets(string1);
answer=(int) strchr(string2, string1);
cout <<"\n string1 <<is at location=" <<ans-(int)string2;
getch();
}

```

آئے اب اس پروگرام کی آٹھ پٹ دیکھتے ہیں۔

```

Before strncat Function
string1 = Yasir and
string2 = Sikandar are best Friends
After strncat Function
string1 = Yasir and Sikandar      , [length] = 18
string2 = Sikandar are best Friends , [length] = 26
strchr(string1, 'd') is at location = 8
Enter string to find from string2: are
are is at location = 9

```

یہ اس پروگرام کی آٹھ پٹ ہے اس میں ہم نے دو اریزی ہیں اور پہلی سڑگ ارے کے آخر پر دوسرا سڑگ ارے کا کچھ حصہ تحریر کروایا ہے۔ اس کے لئے ہم نے strncat(string1, string2, 8) فنکشن استعمال کیا ہے۔ یہ فنکشن سڑگ 2 کے کمکثر سڑگ 1 کے آخر پر لگے گا۔ لہجی آپ جتنی دلیل و درج کریں گے یا بت کر کمکثر تحریر کرے گا۔ اس کے بعد ہم نے ایک فنکشن (strchr()) استعمال کیا ہے۔

```
answer = (int)strchr(string1, 'd');
```

فناش 1 string اے میں سے d کر کیٹھ تلاش کرے گا۔ یہاں پر آپ دیکھیں گے کہ ہم نے ٹاپ کا سٹنگ کی ہے وہ اس لئے کہ یہ جو دلیو ریزن کرے گا وہ نمبر 2 میں ہوگی۔ ہم نے اس کا رزلٹ بھی int ٹاپ کے ویری اسٹبل میں سور کروایا ہے۔ اگر آپ ایسا نہیں کریں گے تو یہ ایردے گا۔

Cannot convert char* to int

اس کے بعد ہم نے انہیں نمبر کو پرنٹ کر دیا ہے۔

cout << "strchr(string1, 'd') is at location=" << ans-(int) string1;
یہاں پر ہم نے answer میں سے سٹرنگ 1 کو تفریق کیا ہے۔ اب سٹرنگ char ٹاپ کا ہے اس کی ٹاپ کا سٹنگ کی ہے اور ایک ایم بات (strchr(فناش صرف کر کیٹھ کو تلاش کرنا ہے۔ اگر آپ سٹرنگ میں سے سٹرنگ کو تلاش کرنا چاہتے ہیں تو اس کے لئے ہم نے یہ لائے کھی

۔

answer=(int) strstr(string2, string1);

اس کے بعد ہم نے ٹاپ کا سٹنگ لی ہے اور اس میں ہم نے string2 میں سے string1 کو تلاش کرنا ہے اور سٹرنگ 1 آپ یوزر سے پوچھ کرے ہیں۔ آپ یہ خود بھی لکھ سکتے تھے مگر ہم نے یہ سہولت یونڈا دی ہے۔ اگر آپ خود لکھنا چاہتے ہیں تو یہ لکھئے۔

answer=(int) strstr(string2, "are");

اب یہ are کو string2 میں سے تلاش کر کے اس کا اندر کی جگہ ریزن کرے گا۔

ایک بات ہم نے یہاں پر ٹاپ کا سٹنگ کی ہے۔ آپ اس کے بعد بھی پروگرام بناسکتے ہیں لیکن اس کے لئے آپ کو پواختہ استعمال کرنا ہو گا۔ پواختہ کیا ہے؟ یہ آپ آگے پڑھیں گے اور اس کی مدد سے آپ ایک پروگرام بنائیں گے۔

مشق

سوال نمبر 1: اریز کیا ہیں اور یہ کس لئے استعمال ہوتی ہیں؟

سوال نمبر 2: ایک ارے بنائیں جس کا سائز 50 ہو اور آپ اس میں صرف 8 ویلوز (Elements) تحریر کریں۔ اس کے بعد یوزر سے ایک ان پٹ لیں اور وہ اس ارے میں اس جگہ تحریر کریں جہاں اس کا درست آرڈر ہو۔ آپ کی ارے کے elements یہ ہونے چاہیے۔

112 , 261 , 272 , 298 , 312 , 391 , 450 , 500

اب فرض کریں کہ یوں 150 نمبر لکھتا ہے تو یہ نمبر 112 اور 261 کے درمیان لکھا جانا چاہئے۔

سوال نمبر 3: ایک ایسا پروگرام تحریر کریں جو یوزر سے سڑنگ ان پٹ لے اور پھر یہ اس کا اٹ آرڈر شو کرے یعنی اگر یوزر awais لکھتا ہے یہ siawa شو کرے اور ایسا ان پٹ لیتھا استعمال کرنا ہے جو سڑنگ میں پسیں کی سہولت فراہم کرتا ہو۔

سوال نمبر 4: ایک ایسا پروگرام تحریر کریں جس میں یوں 3x3 اریز کے لئے ویلوز درج کرے اور پھر بعد میں ان کا مجموعہ اور اوسمط ڈسپلے کروائیں۔ یہ کام آپ فنکشن میں کریں گے اور انہوں ان پٹ 3x3 کی دو اریز یوزر سے لیں گے۔

سوال نمبر 5: ایک ایسا پروگرام بنائیں جو یوزر سے دو سڑنگ لے اور ان کا موازنہ کرے کہ کیا دونوں سڑنگ برابر ہیں یا نہیں اور ان کی لمبائی کیا ہے اور اس کے علاوہ ایک سڑنگ کو دوسرے پر کاپی بھی کروائیں۔

سوال نمبر 6: اس پروگرام کی آٹھ پٹ کیا ہوگی؟

```

int temp[40], count=0, option=0, total=0;
outer:
cout << "Enter Number between[2-40]" ;
cin >>option;
if(option)=2&&option<=40)
{
    for(count=0; count<option; count++)
        cin >>temp[count];
    cout <<"\n";
} //end for loop
count=0;
for(      ; count<option; count++)

```

```
{  
    total=total+temp[count];  
    cout <<"Total is=" .<<total;  
}  
}  
//end if  
else  
{  
    cout <<"\n Wrong Entery:";  
    goto outer;  
}  
getch();  
}  
//end of main;
```

Bsit past papers and books

جوابات

1 : جواب

اے ایسی ڈیٹا ناپ ہے جس میں ایک ہی نام کے وری ایبلر سٹور کئے جاتے ہیں اور ارے میں میوری ایک خاص ترتیب سے اکٹھی جگہ گھیرتی ہے یا ارے ایسے ایجیکٹس کی ترتیب کا نام ہے جن کی ڈیٹا ناپ ایک ہونی چاہئے یہ ایجیکٹس ارے کے اجزاء کھلاتے ہیں۔ آپ اریز اس وقت استعمال کرتے ہیں جو آپ ایک تم کا بہت زیادہ ڈیٹا میوری میں سٹور کروانا چاہتے ہیں۔

2 : جواب

```
#include<conio.h>
#include<iostream.h>
#include<iomanip.h>
void output(int[], int);
void insert(int[], int&, int);
main(void)
{
    clrscr();
    int end=8, temp;
    int array[50]={112,261,272,296,312,391,450,500};
    cout <<"Your Original Array=" <<endl;
    output(array, end);
    cout <<"\n Enter Number:" ;
    cin >>temp;
    insert(array, end, temp);
    cout <<"\n Now number has been inserted:" <<endl;
    output(array, end);
    getch();
}
void insert(int x[], int& temp, int z)
{
    int i=temp;
    for( ;i>0&& x[i-1]>z; i--)
        x[i]=x[i-1];
```

```

x[i]=z;
++temp
}
void output(int temp[], int x)
{
    for (int i=0; i<x-1; i++)
        cout <<temp[i] << ",";
    if(i+1)%12==0
        cout <<endl;
    }
    cout <<temp[x-1] <<endl;
}

```

جواب : ③

```

#include<conio.h>
#include<iostream.h>
#include<stdio.h>
#include<string.h>
void reverse(char[]);
const int size=75;
char temp;
main(void)
{
    clrscr();
    char string[size];
    cout <<"Enter a String:";
    gets(string);           //gets string form user
    reverse(string);
    cout <<"New modified string is:" <<string <<endl;
    cout <<"Enter String:";
    cin.get(string, size);   //cin.get(string,size);is also
                             used to get string
}

```

```

cout << "New modified string is:" <<string <<endl;
getch( );
}

void reverse(char a[])
{
    int length=strlen(a);
    for(int i=0; i<length/2; i++)
    {
        temp=a[i];
        a[i]=a[length-i-1];
        a[length-i-1]=temp;
    }
}

```

جواب :

```

#include<conio.h>
#include<iostream.h>
#include<iomanip.h>
void average(void); //function declaration
int array1[3][3], array2[3][3]; //global array and variables
double sum[3][3];
in i,j;
void main(void)
{
    clrscr();
    cout<< "\n First Matrix: \n";
    cout <<"Enter 3 values in one row then press enter:" <<endl;
    for (i=0; i<3; i++)
        for (j=0; j<3; j++)
        {
            cin >>array1[i][j]; //1st matrix input
            cout <<"\n Second Matrix: \n";
            cout <<"Enter 3 values in one row then press enter:" <<endl;

```

```

for(j=0; j<3; j++)
{
    cin >>array2[i][j];
}                                //2nd matrix input
result( );                      //function calling
average( );
getch( );
}

void result(void)                //function defination
{
    cout <<"\n Sum of Two Matrixes:" <<endl;
    for(i=0; i<3; i++)
    {
        cout <<endl;
        for(j=0; j<3; j++)
        {
            sum[i][j]=array1[i][j]+array2[i][j];
            cout <<setw(8) <<sum[i][j];
        }
    }                                //end of for loop
}                                //end of function
void average(void)                //function definition
{
    cout <<"\n\n Average Matrix:" <<endl;
    for(i=0; i<3; i++)
    {
        cout <<endl;
        for(j=0; j<3; j++)
            cout <<setw(8) <<(sum[i][j]/=2);
    }
}

```

جواب :

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<stdio.h>
void main(void)
{
    clrscr();
    char a[10], a1[15];
    cout <<"\n enter first string:" ;
    gets(a);
    cout <<"\n length of" <<a <<":" <<strlen(a);
    cout <<"\n length of" <<a1 <<":" <<strlen(a1);
    cout <<endl;
    if(strcmp(a,a1)==0)
        cout <<a <<"==" <<a1;
    else
        cout <<a <<"!=" <<a1;
    cout <<"\n copying Strings" <<endl;
    cout <<"\n New First String:" <<strcpy(a,a1);
    getch();
}
```

جواب :

Enter Number between(2-40) 3

Enter 3 values : 6

8

5

Total is : 19

باب نمبر 4

پوائنٹز

اس باب میں آپ پوائنٹز کے بارے میں پڑھیں گے۔ یہ C++ اور C کا ایک اضافی فیچر ہے کیونکہ یہ زیادہ تر کمپیوٹر لینوو بھر میں شامل نہیں ہے مثلاً چاو، پائل، بیسک وغیرہ۔ آپ حق تر ہے بھول گے کہ یہ پوائنٹز کیا ہیں اور یہ کیوں استعمال ہوتے ہیں۔ تو پوائنٹز مندرجہ ذیل کام سر انجام دینے کے لئے استعمال ہوتا ہے۔

ارے عناصر کو ایکسیس کرنا

فتاشن کو آر گومٹ پاس کرنا تاکہ آپ لائیٹ رن نام پر تبدیل کر سکیں

فتاشن کو اریز اور سٹرگنگ پاس کرنا

سشم کی میموری کا ایڈریس معلوم کرنا

ان تمام باتوں کے پیش نظر ہم نے اس باب میں پوائنٹز کا تفصیلی ذرا کیا۔ آپ پوائنٹز سے متعلقہ یہ عنوانات پڑھیں گے۔

ریکٹر اور سٹرگنگ فٹکنٹز	تعارف
شواؤنڈ	ریفرنز
ڈیلیٹ آپریٹر	پوائنٹز
پوائنٹز روپوائنٹز	پوائنٹز ایڈریز
اریز	ریفرنس ریٹن کرنا
مشق	فتاشن پوائنٹز

تعارف:

جب ایک ویری اینبل ڈیکلر کیا جاتا ہے تو اس کے ساتھ تین اہم کلازز منلک ہوتے ہیں مثلاً اس کا نام یعنی ویری اینبل کا نام، اس کی ناپ کیا ہے؟ اور اس کا ایئر لس کیا ہے؟ ایئر لس سے مراد یہ ہے کہ یہ میموری میں کس جگہ شور ہے۔ مثلاً

char temp;

اس ویری استبل کا نام temp ہے۔ اس کی ڈیٹا تاپ کریکٹر ہے اور اس کے علاوہ میموری میں اس ویری استبل کی ویبوکس لوکیشن پر سور ہو گی یعنی اس کا میموری ایڈریس۔ فرض کریں کہ اس کا ایڈریس 1|AB00|2×0 ہے۔ یہ ویبوکس Hexadecimal میں ہے اور آپ کی میموری میں یہ نمبر دیئے گئے ہیں۔ کوئی بھی ویری استبل میموری میں یوں سور ہوتا ہے۔

```
int temp;  
0X1FFOE15  
int [ ]  
temp
```

یہ باکس ویری ایبل کو میموری میں شور کر دے گا۔ یقیناً بتا رہا ہے اس میں باکس کے باس کے بائیں طرف ڈینا تاپ ہے اور باکس کے یونچ ویری ایبل کا نام ہے اور اپر کی طرف ویری ایبل کا میموری میں ایڈریس سے اس کو کوئی ویلوود نہیں ہے۔ اگر آپ اس کو کوئی ویلوود نہیں کرو تو وہ اس باکس میں محفوظ ہو گی۔ مثلاً فرض کریں ہم نے اس میں 12 شور کروا لیا ہے تو وہ اس کے اندر لکھا ہو گا۔ کسی بھی ویری ایبل کی ویلووکس طرح پرنٹ کرواتے ہیں اس سے آپ بخوبی واقف ہیں۔ یعنی cout<<temp; لکھا جائے گا۔ اسی طرح آپ کسی بھی ویری ایبل کا میموری میں موجود ایڈریس بھی معلوم کر سکتے ہیں۔ اس کے لئے 'علامت انتقال کی حاجت ہے اس کو ایڈریس آر یعنی بھی کہتے ہیں اور کسی بھی ویری ایبل کی ایڈریس میں معلوم کرنے کے لئے آپ اس کو یوں لکھتے ہیں۔

```
cout << & temp;
```

آئیے اس کے لئے ایک چھوٹا سا پروگرام لکھتے ہیں۔

مثال نمبر 4.1 دیری اسٹبل کا ایڈر لیں معلوم کرو

```
//header files  
void main(void)  
{  
    int temp=12;  
  
    cout << "Value of temp" << temp;  
    cout << "\n Address of temp=" << &temp;  
    getch( );  
}
```

اب جب آپ اس پر ڈگرام کو ایگزیکوٹ کریں گے تو اس کی آؤٹ پٹ یہ ہو گی۔

Value of temp = 12

Address of temp = 0x1ffacf1d

(References):

ریفرنس:

ایک ریفرنس اصل میں کسی دوسرے ویری ایبل کا ایک مترادف ہوتا ہے اس کو پروگرامگ کی زبان میں کسی دوسری ویرے ایبل کا Alias کہتے ہیں اور آپ کسی بھی ویری ایبل کا Alias کوڈ ڈیکلائر کرتے وقت ہناتے ہیں۔ اس کے لئے اس کے ساتھ ایڈریس آپ پر (&) استعمال کیا جاتا ہے۔ یہ کس طرح کام کرتا ہے آئیے اس کو ایک مثال کی مدد سے سمجھتے ہیں۔

مثال نمبر 4.2 ریفرنس ویری ایبل ڈیکلائر کرنا

```
void main(void)
{
    int temp=21;
    int& ref=temp; //ref is reference for temp
    cout <<"temp=" <<temp <<",ref=" <<ref <<endl;
    temp+=4;
    cout <<"temp=" <<temp <<",ref=" <<ref;
    ++ref;
    cout <<"\n temp=" <<temp <<",ref=" <<ref;
    getch();
}
```

اس پروگرام کی آؤٹ پٹ پر غور کریں کہ اس میں کس بات کی نشاندہی کی جاتی ہے۔

temp = 21 , ref = 21

temp = 25 , ref = 25

temp = 26 , ref = 26

اوپر پروگرام میں ہم نے دو ایڈنیفارز مختلف ناموں کے لئے ہیں لیکن اصل میں یہ دونوں ایک ہی ویری ایبل کے الگ الگ ایڈنیفارز ہیں اور آپ نے آؤٹ پٹ میں دیکھا کہ ہم ایک ایڈنیفارز میں تبدیلی کرتے ہیں اور دوسری ایڈنیفارز خود، بخود تبدیل ہو جاتا ہے اور اسے دو ایڈنیفارز کی ویبیو بائی ڈیفائل آسانی کر دی جاتی ہے۔

آپ چران ہوں گے کہ آخر یہ کس طرح ممکن ہے؟ تو آئیے اس مثال کی مدد سے اس کی لاجک سمجھنے کی کوشش کرتے ہیں۔

مثال نمبر 4.3 ریفرنس ویری ایبل کا ایڈریس معلوم کرنا

```
void main(void)
{
    clrscr();
    int temp=21;
    int& ref=temp;
    cout <<"temp=" <<temp <<"ref=" <<ref;
    cout <<"\n Memory Addresses";
```

```

cout << "\n & temp=" << & temp << ", & ref" << & ref;
getch();
}

```

اب اس کو انگریزی کیوں کریں۔ ہمارے پاس اس کی آڈٹ پٹ کچھ یوں آتی ہے۔

temp = 21 , ref = 21

Memory Addresses

& temp = 0x|FFE04D , ref 0x|FFE04D

آپ کے ذمہ دنوں ایڈنیفارز کا میموری ایڈریس بھی ایک ہی ہے۔ یعنی یہ میموری میں یوں شور ہیں۔

```

int temp;
0X11FFE04D
int 21
int

```

یعنی میموری میں وہی 21 صرف ایک دفعہ نہیں آتی ہے اور temp اور ref کا میموری میں ایک ہی ایڈریس ہے۔ کانسٹانت ایڈنیفارز کی طرح ریفرنس ایڈنیفارز بھی جب ڈیکلر کیا جاتا ہے تو اسے اسی وقت کی شیا زیر کرنا ضروری ہوتا ہے۔

پوائنٹر :

ایک ویری اسٹبل اپنی ویلیو کو ڈائریکٹ ریفرنس کرتا ہے جبکہ ایک پوائنٹر ان ویلیو کو ان ڈائریکٹ ریفرنس کرتا ہے یعنی ویری اسٹبل میں اس کی ویلیو شور ہوتی ہے جبکہ پوائنٹر میں ایک ویری اسٹبل کا میموری ایڈریس ہوتا ہے اور یہ ایڈریس پر متعلقہ ویلیو کی نشاندہی کرتا ہے دوسرے ویری اسٹبل کی طرح پوائنٹر بھی استعمال کرنے سے پہلے ڈیکلر کرنا ضروری ہوتے ہیں۔

اس کو آپ یوں پڑھیں گے کہ یہ ایک فلوٹ ویری اسٹبل کے لئے پوائنٹر ہے یا ایک نریک ویری کے لئے پوائنٹر ہے۔ پوائنٹر کس طرح کام کرتا ہے آئیے اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 4.4 پوائنٹر کا استعمال

```

//header files
void main()
{
    int temp=24;
    int*ptr=& temp;           //ptr holds the address of temp
    cout <<"temp=" <<temp << ", & temp=" << & temp << ", prt="
                                         <<ptr << endl;
    cout <<"& prt=" << & prt;
    getch();
}

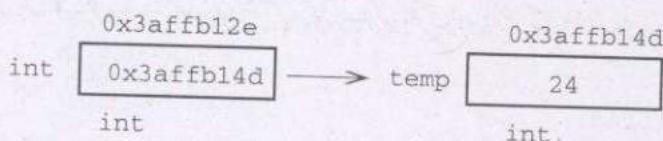
```

اس پروگرام میں ایک دیری اسٹبل `temp` لیا ہے جس کی ولیو 24 ہے اور ایک اٹھ نائپ کا پوائنٹر `ptr` لیا ہے۔ اور اس میں ہم نے `& temp` سوئر کیا ہے اور بعد میں کچھ پرنٹ کروایا ہے جو اس پروگرام کی آٹھ پٹ ہوگی۔

`temp = 24 , & temp=0x3fffb14d , p=0x3afffb14d`

`& ptr=0x3afffb12e`

آپ نے دیکھا کہ `& temp` اور `p` کی ولیو 0x3fffb14d (ایک جیسی) ہے یعنی کہ `temp` اسٹبل کا ایڈریس `ptr` میں سوئر ہے۔ یہ میوری میں کہیں بھی یوں محفوظ ہو گا۔



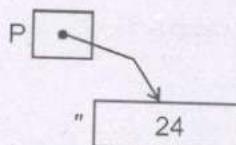
دیری اسٹبل `ptr` کو یہاں ہم پوائنٹر کیلئے کیونکہ اس کی ولیو میوری میں موجود کسی اور ولیو کو پوائنٹ کرتی ہے۔ اس کو ہم `int` پوائنٹر کہیں گے کیونکہ یہ جس ولیو کو پوائنٹ کر رہی ہے وہ اور `ptr` پوائنٹر کی ولیو ایک ایڈریس ہے۔ یہ ایڈریس ہر کمپیوٹر پر مختلف ہو گا یعنی ہمارے پاس جو ایڈریس ہے ہو سکتا ہے آپ کے پاس ایسا ایڈریس نہ آئے۔ اب یہاں پر ووگرام لکھتے ہیں جس میں ہم پوائنٹر کی ولیو ڈالے کروائیں گے۔

مثال نمبر 4.5 پوائنٹر وولیو ڈالے کرنا

```

void main(void)
{
    int temp=24;
    int* ptr=& temp;
    cout << *ptr= " << ptr;
    getch( );
}
*ptr=24
    
```

یہ میوری میں یوں ہو گا۔



ہم ایک بات اور تائیں جو آپ کو پروگرام لکھنے وقت زہن میں رکھنا ہو گی کہ & آپ پر ایک دوسرے کے الٹ ہیں۔ یعنی `p=&n` لکھا جاتا ہے۔ آئیے اس کو ایک مثال سے سمجھتے ہیں۔

مثال نمبر 4.6 ایڈریس آپ پر اور پوائنٹر آپ پر

```

void main(void)
{
    int temp=16;
    int* ptr=& temp;
    
```

```

int& refer=*ptr;
cout <<"ptr=" <<ptr <<endl;
cout <<"refer=" <<refer <<endl;
cout <<"*ptr=" <<*ptr;
getch( );
}

```

اس مثال میں آپ کو یہ بات سمجھانے کی کوشش کی گئی ہے کہ ریفر اور پوائنٹر آپ پر یہ کس طرح اکھا استعمال کرنا ہے۔ اس پروگرام کی آڈٹ پت یہ ہوگی۔

```

ptr = 0x0fff4d1
refer = 16
* ptr = 16

```

پوائنٹر اینڈ اریز:

آپ پوائنٹر پر حسابی آپ پر یہ زندگی لا گر کر سکتے ہیں۔ اس کے علاوہ پوائنٹر میں اضافہ اور کمی بھی کی جاسکتی ہے۔ آپ سوچ رہے ہوں گے کہ پوائنٹر میں میوری کا ایڈریس محفوظ ہوتا ہے پھر یہ سب کیسے ہو سکتا ہے؟ تو آئیے اس کو ایک مثال سے دیکھتے ہیں۔

مثال نمبر 4.7 پوائنٹر ارے کا استعمال

```

void main(void)
{
    const int temp=3; int answer=0;
    int array[size]={20,29,36};
    cout <<"size of(int)=" <<size of(int) <<endl;
    int* team=array*temp-10;
    for(int i=array; i<temp; i++)
    {
        answer+=*i;
        cout <<"\n i=" <<i;
        cout <<"\n *i=" <<*i;
        cout <<"\n Answer=" <<answer;
    }
    cout <<"\n team=" <<team;
    getch();
}

```

اس پروگرام کو ایگزیکیوٹ کرنے کے بعد آپ کو یہ آؤٹ پٹ حاصل ہوگی۔

```
size of(int)=2
i=0x8fb5ffee
*i=20
answer=20
i=0x8fbffff0
*i=29
answer=49
team=0x8fb5ffff2
```

ریفرنس ریٹرن کرنا:

آپ نے فنکشن کے بارے میں تفصیل بیچھا دیا اور اس بات سے بھی واقع ہیں کہ فنکشن کوئی نہ کوئی ولیو ریٹرن بھی کر سکتے ہیں۔ اسی طرح فنکشن کی مدد سے آپ ریفرنس بھی ریٹرن کر سکتے ہیں۔ یہ بھی پروگرامنگ زبان میں ولیو (Lvalue) کہلاتی ہے اور ایسی ولیو کے لئے فنکشن کے لئے لوکل نہیں ہوتی۔

آئیے ایک ایسی مثال لکھتے ہیں جو ریفرنس ریٹرن کرتی ہو۔

مثال نمبر 4,8

```
int action(int& a, int& b)
{
    if(a>b)
        return a;
    else return b;
}
main(void)
{
    int temp, templ;
    cout <<"Enter two values:" ;
    cin >>temp >>templ;
    cout <<temp <<"," <<templ <<"," <<action(temp, templ);
    action(temp, templ)=34
    cout <<endl;
    cout <<temp <<"," <<templ <<"," <<action(temp, templ);
    getch();
}
```

}

اس پروگرام کو ایکزیکوٹ کرنے کے بعد اس کی آؤٹ پٹ کچھ یوں ہو گی۔

Enter two values: 17 21

17 , 21 , 21

17 , 34 , 34

آپ نے اس سے پہلے سڑکر کے شینڈرڈ فنکشنز پڑھے ہیں۔ اس میں ہم نے ایک پروگرام لکھا تھا جو (strchr() اور strstr()) کرتا تھا یہ مثال نمبر 4.9 ہے۔ اس میں آپ کو تاب پ کا سنتگ کی ضرورت پیش آئی جو پروگرام کو ڈنگ کو کافی مشکل بنا رہی تھی۔ اس پروگرام کو آپ پاہنچ کی مدد سے بھی حل کر سکتے ہیں۔ وہ کس طرح آئیے اس کے لئے ایک پروگرام لکھتے ہیں۔

مثال نمبر 4.9: strstr() اور strchr()

```
#include<string.h>
#include<conio.h>
#include<iostream.h>
void main(void)
{
    char str1[]="Failure is another stepping stone to greatness";
    cout <<"string="\n" <<str1 <<"\n";
    char* temp=strchr(str1,'l');
    cout <<"\n strchr(str1, 'l') located at=" <<temp-str1;
    temp=strchr(str1,'y');
    cout <<"\n strchr(str1, 'g') located at=" <<temp-str1;
    temp=strstr(str1, "to");
    cout <<"\n strstr(str1, 'to') located at=" <<temp-str1;
    temp=strstr(str1, "for");
    if(temp==NULL)
        cout <<"\n Sorry Returns Null";
    getch();
}
```

اس پروگرام کی آؤٹ پٹ یہ ہے۔

```
string="Failure is another stepping stone to greatness"
strchr(str1, 'l') located at = 3
```

```
strchr(str1, 'g') located at = 26
```

```
strchr(str1, 'to') located at = 29
```

Sorry Returns Null

اس پروگرام میں ("ا") strchr(temp, 'g') کو سب سے پہلے کال کیا گیا ہے یہ سرگ میں موجود پہلے 1 کو ایک پوائنٹر ریزن کرتا ہے اور ایک پریشن temp-str1 میں اس کریکٹر کا انڈیکس نمبر معلوم کرتی ہے۔ آپ جانتے ہیں کہ ارے صفر (0) انڈیکس سے شارت ہوتی ہے۔ اس کے بعد ("g'") strchr(temp, 'g') کو کال کیا گیا ہے یہ سرگ میں موجود سب سے آخری 0 کو پوائنٹر ریزن کرے گا اور اس کا انڈیکس نمبر ریزن کرتا ہے۔

اس کے بعد ہم نے سرگ کے فناشن کال کیا ہے یعنی (strchr) یہ کریکٹر کا انڈیکس نمبر معلوم کرنے کے لئے استعمال ہوتا ہے۔ آپ اس کے علاوہ سرگ میں سے ایک سرگ کا انڈیکس بھی معلوم کر سکتے ہیں۔ اس کے لئے C++ نے ایک (strstr) کا فناشن استعمال کرنے کی سہولت دی ہے۔ اس لئے فناشن ("to") کو کال سرگ میں سے to کا انڈیکس نمبر ریزن کرے گا۔ یہ انڈیکس نمبر strstr میں سے 2 کا ہو گا۔ یعنی جس کریکٹر سے سرگ شارت ہوتی ہے اس کا یہ انڈیکس نمبر ہوتا ہے۔

اس کے بعد ہم نے strstr(temp, "for") کو کال کیا ہے۔ یہ سرگ میں سے for کو پوائنٹر ریزن کرے گا اور اس کا انڈیکس نمبر ڈھونڈے گا۔

اس کے ساتھ ہم نے az کی کندیش بھی لگائی ہے کہ اگر یہ سرگ میں ہے تو اس کا ملتا تو Null ریزن کرے گا اور وہ پرنس ہو گا۔

فناشن پوائنٹر :

فناشن کے پوائنٹر سے مراد ہے کہ اس کی ذیلفینیشن میں پوائنٹر استعمال کرنا۔ پوائنٹر میموری میں فناشن کا ایڈریس اپنے پاس محفوظ رکھتا ہے اور فناشن پوائنٹر کسی بھی فناشن کو پاس کیا جاسکتا ہے۔ یہ فناشن میں سے ریزن کروایا جاتا ہے۔ آپ سے اریز میں مشور کرو سکتے ہیں اور اس کے علاوہ دوسرے فناشن کے پوائنٹر زکو بھی آسان کر سکتے ہیں۔ آپ نے پیچھے ارے کو ترتیب دینے والوں میں بنا یا تھا۔ آئیے اب پوائنٹر فناشن کی مدد سے یہ پروگرام حل کرتے ہیں۔

مثال نمبر 4.10 فناشن پوائنٹر

```
#include<iomanip.h>
#include<conio.h>
#include<iostream.h>
void sorting(int[], const int, int(*) (int, int));
int ascend(int, int);
int descend(int, int);
void main(void)
{
    int temp, count;
    int array[ ];
    cout << "Enter any 12 Numbers" << endl;
    for(int i=0; i<12; i++)
        cin >> array[i];
```

```
cout <<"Enter 2 to sort in ascending order:";  
cout <<"\n Enter 1 to sort in descending order:";  
cin >>temp;  
cout <<"here is original array: \n";  
for(count=0; count<12; count++)  
cout <<array[count] <<setw(5);  
if(temp==1)  
{  
    sorting(array ,12, descend);  
    cout <<"\n Array elements in Descending order. \n";  
}  
else  
{  
    sorting(array ,12, ascend);  
    cout <<"\n Array elements in Ascending order. \n";  
}  
for (counter=0; counter<12; counter++)  
cout <<array[counter] <<setw(5);  
getch();  
  
void sorting(int l, const int s, int(*match)(int ,int))  
{  
    void arrange(int*, int*);  
    for(int i=1; i<s; i++)  
        for(int j=0; j<s; j++)  
            if(!(*match)(q[j], q[j+1]))  
                arrange(& q[i], & q[j+1]);  
}  
void arrange(int* data1, int* data2)  
{  
    int temp;  
    temp=*data1;  
    *data1=*data2;
```

```

    *data2=temp;
}

int ascend(int x, int y)
{
    return y<x;
}

int descend(int a, int b)
{
    return b>a;
}

```

اس پروگرام کو ایگر لایکیوٹ کریں گے تو یہ آؤٹ پٹ سکرین پر ڈالے ہوگی۔

Enter any 12 Numbers

1 19 2 12 6 8 45 96 20 16 82 12 90

Enter 2 to sort in Ascending order:

Enter 1 to sort in Descending order: 1

Here is original array

1 19 2 12 6 8 45 96 20 16 82 12 90

Array elements is Descending order

96 90 82 45 20 19 16 12 8 6 2 1

اب آپ دوسری دفعہ پروگرام ایگر لایکیوٹ کرتے ہیں اور اس دفعہ آپ آؤٹ پٹ Ascending آرے کھانا چاہتے ہیں تو 2 تحریر کجھے گا۔

ث

Enter any 12 Numbers

0 99 1 82 64 56 12 28 6 72 8 81

Enter 2 to sort in Ascending order:

Enter 1 to sort in Descending order: 2

Here is original array

0 99 1 82 64 56 12 28 6 72 8 81

Array elements is Ascending order

0 1 6 8 12 28 56 64 72 81 82 99

کریکٹ اور سڑنگ فنکشن:

آپ نے پہلے باب میں بھی سڑنگ ہیڈر فائل میں موجود فنکشن کے بارے میں پڑھا تھا۔ یہاں ہم پوائنٹر کی مدد سے آپ کو ان فنکشن کے بارے میں بتائیں گے۔ نیچے ان کا نیبل درج ہے۔

فناشن وضاحت	فناشن پروٹوٹاپ
یہ سرنگ s2 کو s1 میں سینور کرتا ہے اور s1 کی ولیووریٹن کرتا ہے	char *strcpy(char *s1, char *s2)
یہ سرنگ c2 کے تین لفظ c1 ارے میں کاپی کرتا ہے	char *strncpy(char *c, char *c1, 3)
یہ سرنگ c2 کو c1 کے آخر پر لکھتا ہے c1 ارے کا آخری کریکٹر (Null) c2 کے پہلے کریکٹر سے تبدیل ہو جائے گا	char *strcat(char *c, char *c1)
یہ سرنگ c2 کے 5 کریکٹر (لفظ) c1 ارے میں لکھتا ہے	char *strncat(char *c, char *c1, 5)
یہ سرنگ c1 کا c2 سے موازنہ کرتا ہے کیا دونوں سرنگز برابر ہیں یا نہیں اگر برابر ہوں گے تو 1 ورنہ 0 ریٹن کرتا ہے	int strcmp(char *c, char *c1)

یہ سرنگز کے کچھ اہم فناشنیں فصل تھی۔ اس سے پہلے آپ اس باب میں بھی سرنگز کے دو فناشن کو پوائنٹر کی مدد سے استعمال کرنے کا عمل پڑھ پچھے ہیں۔ آئیے اب ایک اور پروگرام لکھتے ہیں۔

مثال نمبر 4.11 سرنگ اور فناشن کا استعمال

```
void main(void)
{
    clrscr();
    char *str1 = "Hello Sikandar";
    char *str2 = "How are you";
    char *str3 = "Hello Sikandar";
    cout << "str1=" << str1 << "\n"
        << "str2=" << str2 << "\n"
        << "str3=" << str3 << "\n";
    cout << "strcpy(str1, str2)=" << strcpy(str1, str2);
    cout << "\n strncpy (str2, str3, 4)=" << strncpy (str2, str3, 4);
    cout << "\n";
    cout << "Length: \n strlen(str1)=" << strlen(str1);
    getch();
}
```

آپ نے دیکھا کہ ہم نے پروگرام میں تین کریکٹر پوائنٹر ایمنی شیلائیز کئے ہیں اور بعد میں ان کا موازنہ کیا ہے۔ اور ان میں سے str1 کی لمبائی معلوم کی ہے۔ جب آپ اس پروگرام کو ایگزیکیوٹ کریں گے تو یہ پروگرام یہ آٹھ پڑھ سلے کرے گا۔

```
str1 = Hello Sikandar
str2 = How are you
```

```

str3 = Hello Sikandar
strcpy(str1, str2) = How are you
strncpy(str2, str3, 6) = Hello e you
Length:
strlen(str1) = 11

```

(New) نیوا آپ پریٹر:

آپ نے پوائنٹر ڈیکلر کرنے والے اور اس کے استعمال سے بھی بخوبی واقف ہوں گے۔ اب ہم آپ کو تھوڑا سا ایڈوالس لے کر جلتے ہیں۔ آپ پوائنٹر ایسے ڈیکلر کرتے ہیں۔

```
int *ptr;
```

اس کا مطلب ہے کہ ptr ویری ایڈوال int کے لئے یاد پوائنٹر ہے اسے انگلش میں یوں لکھتے ہیں۔

```
ptr is a pointer to an integer
```

یہ صرف پوائنٹر کے لئے میموری متعین کرتا ہے اور پوائنٹر کو ویلوں کی میموری ایڈوالیں دیکھیں کہ اس ایڈوالیں پر بھی تک کوئی میموری متعین نہیں کی گئی۔ یعنی ptr ایسی شیلائیز نہیں کیا گیا اور یوں یہ کسی بھی میموری کو ویلوں کی ویلوں کو پوائنٹ کرنے کی بھی نہیں کر رہا۔ اب اگر آپ یوں لکھتے ہیں۔

```
*ptr = 81; //Error message
```

تو یہ ایر ہو گا یعنی یہ جس میموری کو پوائنٹ کر رہا ہے اگر آپ اسے ایکسیس کرے تو اس کو ویشوں کریں گے تو یہ ایک ایر ہو گا کہ پوائنٹر ptr کے لئے کسی تم کا ڈیتا شور نہیں کیا گیا۔ ہم اس باب کے آغاز میں ایک بات واضح کر چکے ہیں کہ جب ڈیکلر کیا جاتا ہے تو یہ اسی وقت ایسی شیلائیز بھی کر دیا جاتا ہے یعنی اس ایر سے پہنچ کا بہترین طریقہ یہ ہے کہ آپ اسے اسی وقت ایسی شیلائیز کرو ایں۔ بچ پوائنٹر ڈیکلر کیا جائے۔

```
int* temp = 171; //temp = 171
```

```
int* ptr = & temp //ptr = address of temp
```

اب اگر آپ ptr * کو ایکسیس کرتے ہیں۔

```
cout << *ptr;
```

تو اس میں کوئی پر الٹر نہیں ہے کیونکہ temp کا میموری ایڈوالیں ptr میں شور کیا جا چکا ہے۔ اس کے علاوہ اس کا ایک اور طبقہ ہے جس کی مدد سے آپ اور والا کام جا سانی کر سکتے ہیں۔

```
int* prt;
```

```
ptr = new int;
```

```
*ptr = 171;
```

یہ پہلی دوالہ آپ کہاں بھی لکھ سکتے ہیں۔

```
int* ptr = new int;
```

آپ پریٹر کا فنکشن ہے کہ یہ میموری میں ایسی بائنس کا ایڈوالیں ریٹرن کرتا ہے جو کسی اور ویری ایڈوال کے لئے بھی تک ریزرو نہ کی گئی ہوں۔ new اب جب آپ نے int ptr=new کھا ہے تو اس کا یہ مطلب ہے کہ int کے لئے دو بائنس ریزرو کر دے یعنی اسے بائنس کا ایڈوال دے دیا گیا ہے اور یہ اس بات کی وضاحت بھی کرتا ہے کہ prt اس وقت کسی اور ویری ایڈوال کے استعمال میں نہیں ہے۔ ایک اور بات کہ جب آپ new آپ پریٹر کے

ذریعے کسی پوائنٹر دیری اس بدل کو اینی شیلائیز کرتے ہیں تو وہ پوائنٹر اینی شیلائیز ہو گا نہ کہ وہ میموری جس کو وہ پوائنٹ کر رہا ہو گا۔ ایسا ایک صورت میں ممکن ہے اگر آپ تمام کام ایک ہی لائن میں مکمل کریں۔ مثلاً

```
int* ptr = new int(171); //both and * ptr have been initialized
```

ڈیلیٹ (Delete) آپریٹر:

یہ آپریٹر new آپریٹر کا مقابلہ ہے یعنی یہ اس کامل تبدیل کر دیتا ہے مثلاً آپ new کی مدد سے کسی پوائنٹر کے لئے میموری متعین کرتے ہیں اور ڈیلیٹ کی مدد سے آپ اس میموری کو پھر سے آزاد یعنی خالی کرو سکتے ہیں۔ اس سے یہ واضح ہوا کہ آپریٹر میموری کو خالی کروانے کے لئے استعمال ہوتا ہے۔ مثلاً طرح استعمال کیا جانا چاہئے آئیے ایک نظر دیکھتے ہیں۔

```
float* ptr = new float(7.219);
delete ptr;
cout << *ptr; //Error ptr has been deallocated
```

جب ایک دفعہ آپ کسی بھی پوائنٹر جیسی ایبل کو deallocate کر دیتے ہیں تو یہ اس وقت تک دوبارہ استعمال نہیں کیا جاسکتا جب تک کہ reallocate کیا جائے۔ ایک dangling pointer ہو اپنے پوائنٹر بھی کہلاتا ہے۔ ایک بات اور جو قابل غور ہے کہ delete آپریٹر اس وقت استعمال کیا جاسکتا ہے جب آپ کسی پوائنٹر کو new آپریٹر کی مدد سے اینی ٹیزر کرتے ہیں اور آپ کا نشست پوائنٹر کو deallocate کر سکتے ہیں۔ مثلاً

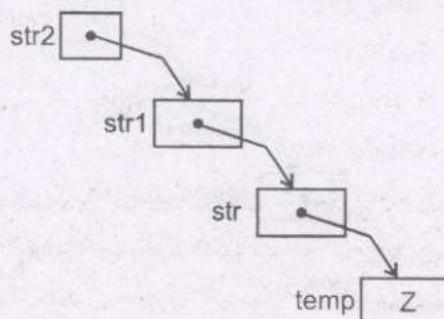
```
const int* ptr = new int;
delete ptr; //Error cannot delete pointer to const
```

پوائنٹر زٹو پوائنٹر:

اس سے پہلے تک ہمارا ہر پوائنٹر میموری ایڈریس یا ولیو کو پوائنٹ کرتا تھا لیکن ایسا پوائنٹر کسی دوسرے پوائنٹر کو بھی پوائنٹ کر سکتا ہے۔ مثلاً

```
char temp = 'Z';
char* str = & temp;
char** str1 = & str;
char*** str2 = & str1;
****str2 = 'S';
```

یہ میموری میں کچھ یوں سور ہوں گے۔



اریز:

آپ اریز سے بخوبی واقف ہیں لیکن یہاں پر ہم آپ کو اریز کے بارے میں مزید کچھ بتائیں گے۔ آپ نے اب تک ایسی اریز بنائی ہیں جو کہ کپائل نائم پر بنتی ہیں اور اس کی میموری اسی وقت ویری اسٹبلر کی لئے متعین کر دی جاتی ہے۔ ایسی اریز کو static ارے کہتے ہیں۔ آپ اس کے علاوہ ایسی اریز بھی بناسکتے ہیں جو رون نائم پر بنتی ہیں اور اس کی میموری ویری اسٹبلر کے لئے اس وقت متعین کی جاتی ہے جب اس ارے کی ذیکری ریشن ایگز کیوٹ ہو۔ ایسی اریز کس طرح بنائی جاسکتی ہیں اور یہ کس طرح کام کرتی ہیں۔ آئیے اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 4.12 ارے dynamic

```

void input(double*&, int&);

void output(double*, int);

void main(void)
{
    clrscr();
    double* temp;
    int q;
    input(temp, q);
    output(temp, q);
    delete[] temp;
    input(temp, q);
    output(temp, q);
    delete[] temp;
    getch();
}

void input(double*& temp, int& q)
{
    cout << "\n How many values you want to Enter:" ;
    cin >> q;
    temp=new double[q];
    cout << "\n Enter values one per one line: \n";
    for(int i=0; i<q; i++)
    {
        cout << i+1 << ":" ;
        cin >>temp[i];
    }
}

```

```
void output(double* temp, int q)
{
    for(int i=0; i<q; i++)
        cout <<temp[i] << " ";
}
```

اس پروگرام میں ہم نے دو فکشنز بنائے ہیں۔ ایک یوزر سے اریز کی ویبیوز لیتا ہے جبکہ دوسرا وہی ویبیوز پرنٹ کرواتا ہے۔ اس پروگرام میں ہم ایسی ارے بنارے ہے جو ان تابع پر ایگزکیوٹ ہو گی اور یوزر سے پڑھتے گی کہ وہ کتنی ویبیوز ارے میں درج کرنا چاہتا ہے۔ آئیے اس پروگرام کی آڈٹ پٹ دیکھتے ہیں۔

How many values you want to Enter: 4

Enter one value per one line

```
1 : 12
2 : 36
3 : 91
4 : 56
12 36 91 56
```

How many values you want to Enter: 2

Enter one value per one line

```
1 : 102
2 : 371
102 371
```

مشق

سوال نمبر 1: پوائنٹر ز کیا ہیں اور یہ کس لئے استعمال کئے جاتے ہیں؟

سوال نمبر 2: ایک ایسا پروگرام بنائیں جس میں ایک فنکشن کو اے پاس کی گئی ہو؟ اس کے لئے آپ کو پوائنٹر استعمال کرنا ہوگا۔

سوال نمبر 3: ایک ایسا پروگرام بنائیں جو یوزر سے دو شرکز لے اور ان کا موازنہ کرے کہ کیا وہ برابر ہیں یا نہیں۔ اس میں آپ نے `strcmp` فنکشن استعمال نہیں کرنا بلکہ پوائنٹر اور یوزر دینا اس فنکشن استعمال کرنا ہے۔

سوال نمبر 4: آپ نے مشہور سیریز $f(n) = f(0) + f(1) + f(2) + \dots + f(n-1)$ کے بارے میں پڑھا ہوگا۔ اس کو کیلکولیٹ کرنے کے لئے ایک پروگرام تحریر کریں جس میں پوائنٹر کا استعمال ہو اور جو فنکشن بنائیں اس کو پوائنٹر پاس کیا گیا ہو۔ مثلاً یہ سیریز یوں عمل کرتی ہے۔

$$1 + 3 + 6 + 11 = 21$$

سوال نمبر 5: ڈیلیٹ اور `new` آپ پر یہ میں کیا فرق ہے؟

سوال نمبر 6: پروگرامنگ کے ان کوڈز میں کیا ایرہ ہے؟

- (i) `int* compare;`
`int* temp = &compare;`
- (ii) `char a = 'z';`
`char b = &a';`
- (iii) `int i = new int;`
`int* j = new int`
- (iv) `int b[20];`
`for(int i=0; i<20; i++)`
`*b++ = i*i;`

جوابات

① : جواب

ایک ویری اسٹبل اپنی ویلیو کوڈ ائریکٹ ریفرنس کرتا ہے جبکہ ایک پوائنٹر اپنی ویلیو کو ان ڈائریکٹ ریفرنس کرتا ہے لیکن ویری اسٹبل میں اس کی ویلیو شور ہوتی ہے جبکہ پوائنٹر میں ایک ویری اسٹبل کا میموری ایڈریس ہوتا ہے اور یہ ایڈریس آپ کی اس ویلیو کی نشاندہی کرتا ہے۔ دوسرے ویری اسٹبل کی طرح پوائنٹر زیبھی استعمال نہ سے پہلے ڈیکلیر کرنا ضروری ہوتے ہیں۔

② : جواب

```
#include<conio.h>
#include<iostream.h>
#include<iomanip.h>
void resultant(double*);
const int size=6;
main(void)
{
    clrscr();
    double array[size]={12,15,91.5,56,32.6,78};
    cout <<"Your original array" <<endl;
    for(int i=0; i<size; i++)
        cout <<array[i] <<"," ;
    cout <<endl;
    resultant(array);
    for(int i=0; i<size; i++)
        cout <<"Array[" <<i <<"]=" <<array[i] <<endl;
    getch();
}
void resultant(double* temp)
{
    for(int i=0; i<size; i++)
        *temp++*=3;
```

جواب : 3

```

#include<conio.h>
#include<iostream.h>
#include<stdio.h>
int relastic(const char*, const char*);
main(void)
{
    clrscr();
    int compare;
    char str1[75], str2[75];
    cout <<"Enter 1st String:";
    gets(str1)
    cout <<"\n Enter 2nd String:";
    gets(str2)
    compare = relastic(str1, str2);
    if(compare==1)
        cout <<str1 <<"!=" <<str2;
    else
        cout <<str1 <<"==" <<str2;
    getch();
}
int relastic(const char *n1, const char *n2)
{
    //while(*n1 != '\0')
    //++n1;
    for(; *n1 != '\0' && *n2 != '\0'; n1++, n2++)
        if(*n1 != *n2)
            return 0;
    return 1;
}

```

4: جواب

معلوم کرنے کے لئے یہ پروگرام بنائیں۔
 $f(0)+f(1)+f(2)+\dots+f(n)$

```

int result(int(*)(int), int);
int dem(int);
void main()
{
    clrscr();
    int val;
    cout << "Enter Number of values to be Calculated";
    cin >> val;
    cout << result(dem, val);
    getch();
}

int result(int(*temp)(int a), int b)
{
    int ans=0;
    for(int i=0; i<=b; i++)
        ans+=(*temp)(i);
    return ans;
}

int dem(int a)
{
    return a*2;
}

```

5: جواب

new کی ورڈ کسی بھی پوائنٹر کے لئے میموری ریزرو کرتا ہے۔ یہ میموری میں ایسی بائس کا ایڈر لیس ریزرن کرتا ہے جو کسی اور دیری اہل کے لئے بھی تک ریزرو نہ کی گئی ہوں۔

delete کی ورڈ new آپریٹر کے الٹ ہے۔ یہ **new** کی مدد سے پوائنٹر کے لئے ریزرو کی ہوئی میموری کو ختم کرتا ہے۔ یعنی **delete** کی مدد سے آپ **new** کی مدد سے ریزرو کی ہوئی میموری کو پھر سے آزاد (خالی) کرو سکتے ہیں۔

6 : جواب

(i) Error

cannot convert 'int**' to 'int*'

(ii) Error

cannot convert char* to char

(iii) Error

cannot convert int* to int

(iv) Error

value required

باب نمبر 5

سٹرکچر اینڈ کلاسز

ایک سٹرکچر سادہ ویری اسٹبلر کا جمود ہوتا ہے اور یہ ویری اسٹبلر کی بھی ناپ کے ہو سکتے ہیں اور سٹرکچر میں شامل اجزاء سٹرکچر کے ڈیٹا ممبر کہلاتے ہیں۔ درحقیقت سٹرکچر بنانے کا طریقہ تقریباً کلاسز سے ملتا ہے۔ فرق صرف یہ ہے کہ سٹرکچر ڈیٹا کا جمود ہوتا ہے جبکہ کلاسز میں ڈیٹا اور فنکشن دونوں شامل ہوتے ہیں اور کلاسز کان فنکشن کو ممبر فنکشن کہتے ہیں۔ آپ فنکشن کے بارے میں تفصیل سے پڑھ پکے ہیں اور اس باب کے پہلے حصہ میں آپ سٹرکچر کے بارے میں پڑھیں گے اور بعد میں آپ کلاسز کے بارے میں معلومات حاصل کریں گے۔

اس سے پہلے آپ نے جو چیزوں پر تلاوة اور بحیث اور بینڈ ڈیزائن (OOD) کہلاتا ہے اور اس باب میں اور بحیث اور بینڈ پروگرامنگ (OOP) کے بارے میں پڑھیں گے۔ یہ C++ میں پروگرامنگ کا یونٹ کلاس ہوتا ہے جس کا بعد میں اور بحیث بنایا جاتا ہے۔ اس باب میں آپ ممبر فنکشن دونوں کے بارے میں تفصیل سے پڑھیں گے۔

پرائیویٹ اور پیبل کی ورڈز



سٹرکچر



کنسٹرکٹر



سٹرکچر اجزاء کو ایکسیس کرنا



کنسٹرکٹر اور لوڈنگ



سٹرکچر پوائزٹر



او بحیث بطور آر گومنش



سٹرکچر Nested



فنکشن کے اور بحیث ریٹن کرنا



یوزر ڈیفائل سند ڈیٹا ناپس



ڈیٹرکٹر



کلاسز



مشن



کلاس ڈیکلریشن



سٹرکچر:

سٹرکچر ویری اینڈ لارن کے مجموعہ کو کہتے ہیں اور سٹرکچر میں شامل ویری اینڈ کسی بھی ناٹپ کے ہو سکتے ہیں۔ مثلاً یہ ویری اینڈ int ناٹپ کے بھی ہو سکتے ہیں اور ان میں سے بعض double ناٹپ کے بھی ہو سکتے ہیں۔ سٹرکچر میں شامل ڈیٹا آئیٹمز (ویری اینڈ وغیرہ) سٹرکچر کے ممبر (ارکان) کھلاتے ہیں۔ آپ کسی بھی قسم کا سٹرکچر یوں بناتے ہیں۔

```
struct employee
{
    int emp;
    float temp;
    char choice;
};
```

جب بھی آپ سٹرکچر بنانا چاہتے ہیں تو اس کے لئے کوئی دوسرے struct employee اس کا استعمال کریں گے۔ ایڈنٹیفایر سٹرکچر ٹائگ ہے یا یہ سٹرکچر کا نام ہے اور یہ سٹرکچر ناٹپ کے ویری اینڈ ڈیلکلیر کرنے کے لئے استعمال ہوتا ہے۔ سٹرکچر بریکلیش ({{}}) کے اندر جو نام ڈیلکلیر کئے گئے ہیں وہ سٹرکچر ممبر ہیں۔ ایک اہم بات یہ ہے کہ سٹرکچر کی ڈیلکلیر میں جو ویری اینڈ ڈیلکلیر کے جائیں گے ان کے نام منفرد ہونا ضروری ہیں اور سٹرکچر کا اختتام ہمی کالن (;) سے ہونا ضروری ہے۔ ایک سٹرکچر خود اپنا انسٹینس (instance) نہیں بنائتا اس رہ انسٹینس کیا ہے آپ آئے گے تفصیل سے پڑھیں گے۔ فی الحال آپ صرف یہ خیال رکھیں کہ آپ سٹرکچر کے نام کا ایڈنٹیفایر سٹرکچر کی باڈی میں ڈیلکلیر ٹائگ لے سکتے۔ یعنی employee سٹرکچر ممبر کا نام employee نہیں ہو سکتا۔

سٹرکچر اجزاء (ارکین، ممبر یا ویری اینڈ) کو ایکسیس کرنا:

آپ نے اوپر سٹرکچر کے ممبر ڈیلکلیر کئے ہیں اب آپ اگر ان کو سٹرکچر کی باڈی سے باہر ایکسیس کرنا چاہتے ہیں اس کے لئے آپ کو ممبر ایکسیس آپریٹر استعمال کرنا ہو گا۔ سٹرکچر کے لئے ممبر ڈیلکلیر کرنے کے لئے دو ایکسیس آپریٹر کے استعمال کئے جاتے ہیں۔ لاءِ آپریٹر (.) اور ایرو آپریٹر (→)

ڈاٹ آپریٹر کی مدد سے آپ سٹرکچر کے ممبر (ویری اینڈ) کو ایکسیس کرتے ہیں۔ یعنی اس کی مدد سے آپ او جیکٹ کا یہ اینڈ ایڈنٹیٹل نام یا او جیکٹ کارینفس ایکسیس کرتے ہیں۔ مثلاً ہم نے employee کا او جیکٹ employee کا ایکسیس کرنا چاہتے ہیں تو اسے یوں ایکسیس کر سکتے ہیں۔

```
cout <<emple.temp;
```

ایرو آپریٹر آپ اس وقت استعمال کریں گے جب آپ نے سٹرکچر ممبر کو ایکسیس کرنے کے لئے او جیکٹ کو پاؤ اسٹر ڈیلکلیر کیا ہو۔ فرض کریں کہ *emplptr پاؤ اسٹر ڈیلکلیر کیا گیا ہے جو employee او جیکٹ کو پاؤ اسٹر کرتا ہے۔ تو جب آپ اس کی مدد سے سٹرکچر ممبر کو ایکسیس کرنا چاہیں گے تو ایرو آپریٹر لکھا کس طرح جاتا ہے۔ اس کے لئے تفریقی علامت (minus operator) اور اس کے بعد (greater) علامت لکھیں اور ان کے درمیان پسیں نہیں ہونی چاہئے۔ بھی ہو سکتا ہے کہ یہ آپ کے لئے کنیوزنگ پاؤ اسٹر ہو یعنی جب آپ اس کی مثال دیکھیں گے تو یہ کلستر ہو جائے گا۔

سٹرکچر کس طرح بنایا جاتا ہے اور کس طرح اس کے ڈیٹا ممبر کو ایکسیس کیا جاسکتا ہے آئیے اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 5.1 سُتر کچر بنانا

```

struct student
{
    char name[20], & course[5];
    int rollno;
};

void main(void)
{
    clrscr();
    student st1, st2;      //variable of new type student
    cout << "\n Enter Student1 Name:" ;
    cin >>st1.name;
    cout << "\n Enter Student1 class:" ;
    cin >>st1.course;
    cout << "\n Enter Student1 rollno:" ;
    cin >>st1.rollno;
    cout << "\n Enter Student2 Name:" ;
    cin >>st2.name;
    cout << "\n Enter Student2 class:" ;
    cin >>st2.course;
    cout << "\n Enter Student2 rollno:" ;
    cin >>st2.rollno;
    getch();
}

```

اس پروگرام میں ہم نے ایک student کے نام سے سُتر کچر بنایا ہے اور اس میں تین ویری اسٹبلوڈیکلیر کئے ہیں۔ اس کے بعد ہم نے main میں سُتر کچر ناپ کے دو ویری اسٹبلوڈیکلیر کئے ہیں۔

```
student st1, st2;
```

آپ ان کو سُتر کچر کے او بھیکٹ بھی کہہ سکتے ہیں۔ ان کی مدد سے ہم سُتر کچر کے ڈیتا مبرزا ایکسیس کریں گے۔ اس لائن سے یہ بھی ظاہر ہوا کہ آپ ایک سُتر کچر کے کئی او بھیکٹ بھی بناتے ہیں یا سُتر کچر ناپ کے کئی ویری اسٹبلوڈیکلیر کر سکتے ہیں۔ بعد میں ہم نے student1 کے متعلق ڈیتا یوزر سے حاصل کیا ہے اور اس کے لئے ہم نے ویری اسٹبل1 استعمال کیا ہے جبکہ student2 کا ڈیتا حاصل کرنے کے لئے st2 استعمال کیا ہے۔ یہ ایک بالکل سادہ سا پروگرام ہے جو صرف آپ کو یہ بات واضح کرتا ہے کہ کس طرح سُتر کچر بناتا ہے اور کیسے اس کو بعد میں استعمال کیا جاسکتا ہے اور یہ او بھیکٹ

پروگرام کی طرف پہلا قدم ہے کہ ہم نے ایک نام کے دو اوبجیکٹ بنائے ہیں اور پھر وہ آگے استعمال کئے ہیں۔ اس پروگرام میں ہم نے ہر شوٹن کے لئے الگ سے ان پٹ لی ہے۔ یہاں پر صرف دوری کارڈز کی ہمیں ضرورت تھی لیکن اگر آپ کو فرض کریں 25 ریکارڈز چاہیں تو کیا ہر ایک کے لئے الگ سے ان پٹ لیں گے؟ تو ایسا نہیں ہے اس کے لئے ہمارے کا استعمال کریں گے اور لوپ کی مدد سے ان پٹ لیں گے۔ آئیے اس کے لئے ایک ستر کچو پروگرام لکھتے ہیں۔

مثال نمبر 5.2 ریکارڈ ملاش کرنا

```

struct student
{
    char name[20], aclass[5], address[40];
    int rollno;
};

void main(void)
{
    clrscr();
    student st[3];
    int count, tryno;
    for(count=1; count<=3; count++)
    {
        cout << "\n Enter student" << count << "Rollno:" ;
        cin >>st[count].rollno;
        cout << "\n Enter student" << count << "Name:" ;
        cin >>st[count].name;
        cout << "\n Enter student" << count << "Class:" ;
        cin >>st[count].aclass;
        cout << "\n Enter student" << count << "Address:" ;
        cin >>st[count].address;
    }
    cout << "\n Enter Rollno to find student:" ;
    cin >>tryno;
    for(count=1; count<=3; count++)
    {
        if(tryno==st[count].rollno)
        {

```

```

cout << "student rollno:" << st[count].rollno;
cout << "\n student name:" << st[count].name;
cout << "\n student class:" << st[count].aclass;
cout << "\n student address:" << st[count].address;
}
}
getch();
}

```

اس پروگرام میں ہم نے ایک شوڈنٹ نام کا سٹرکچر بنایا ہے اور اس میں چار ویری اینڈ ڈیٹائلر کئے ہیں۔ (main میں شوڈنٹ کا او بجکٹ بنایا ہے اور یہ ایک ارے ناپ کا او بجکٹ ہے جس کو ویزو پاس کی گئی ہے۔ for کے لوپ کی مدد سے ہم نے ہر شوڈنٹ کے لئے ان پٹ لی ہے یعنی اس کا یہ فائدہ ہے کہ ہم نے خواہ کتنے ریکارڈ یوزر سے کیوں نہ لئے ہوں، ہم صرف ایک شوڈنٹ کی کوڈنگ کرتے ہیں اور for لوپ کی مدد سے اسے کنٹرول کرتے ہیں۔ اس کے بعد ہم نے یوزر سے ایک اور ان پٹ لی ہے۔

```
cin >> tryno;
```

اب یہاں یوزر جو بھی نمبر تحریر کرے گا اس کو اپر لکھ گئی ریکارڈ میں موجود روٹنگر کے ساتھ ملایا جائے گا۔ اگر یہ نمبر مل گیا تو اس سے مختلف ڈیٹا شو کر دیا جائے گا۔ اب یہ شوڈنٹ کے روٹنگر کے برابر ہے یا نہیں؟ اس کے لئے ایکسٹ میٹ استعمال کی گئی ہے۔

```
if (tryno == st[count].rollno)
```

یہ ایکسٹ میٹ tryno کو ہر روٹنگر کے ساتھ ملائے گی اور یہاں کوئی نامہ نمبر کی ریکارڈ سے ملتا ہے یا نہیں۔

سٹرکچر پوائنٹر :

ہم نے اس باب کے آغاز میں آپ کو بتایا تھا کہ آپ ایک سٹرکچر کو پوائنٹر بھی پاس کر سکتے ہیں اور ایسے سٹرکچر کے ویری اینڈ ڈیٹائلر کو ایکس کرنے کے لئے پوائنٹر آپریٹر استعمال کیا جاتا ہے۔ آپ یہ کام کس طرح پر فارم کر سکتے ہیں آئیے اس پر فارم کی مدد سے بھجھنے کی کوشش کرتے ہیں۔

مثال نمبر 5.3 سٹرکچر پوائنٹر

```

struct show
{
    char name[20], sex[6];
    int age;
};

show* inpt(void);
void output(show* );
void main(void)
{
    clrscr();

```

```

show* sh;
sh=input( )
output(sh);
getch();
}

show* input(void)
{
    show* temp;
    cout <<"\n Enter your Age:" ;
    cin >>temp->age;
    cout <<"\n Enter your Name:" ;
    gets (temp->name);
    cout <<"\n Enter your Sex:" ;
    gets (temp->sex);
    return temp;
}

void output(show* temp)
{
    cout <<"\n Your Age:" <<temp->age;
    if(temp->age<=10)
        cout <<"\n You are a child" <<temp->sex;
    else
        if(temp->age<=20)
            cout <<"\n You are a teenage" <<temp->sex;
        else
            if(temp->age<=30)
                cout <<"\n You are an adult" <<temp->sex;
            else
                cout <<"\n You should now take rest: an old" <<temp->sex;
}

```

آپ جب اس پروگرام کو رن کریں گے تو آٹھ پٹ یہ ہو گی۔

Enter you Age: 22

visit our website: www.pupapersbook.blogspot.com

Your Name: Yasir Aurangzaib

Your Sex: Male

Your Age 22

You are an adult male

اس پروگرام میں ہم نے ایک سٹرکچر `show` بنایا ہے اور دو فنکشن لکھے ہیں۔ ان میں سے ایک فنکشن (`input`) سٹرکچر پوائنٹر ویلیور ریٹن کرتا ہے جبکہ دوسرا فنکشن (`output`) کو سٹرکچر پوائنٹر بطور پیرا میٹر پاس کیا ہے۔ اب آپ سوچ رہے ہوں گے کہ پوائنٹر سٹرکچر کس طرح پاس کیا جاتا ہے تو اس کا حل (`main`) میں موجود ہے۔ ہم نے سٹرکچر کا او بجیکٹ `sh` بنایا ہے اور یہ پوائنٹر او بجیکٹ ہے پھر اس کو ایک فنکشن میں سے پاس کر دیا ہے جبکہ دوسرے فنکشن کو اس او بجیکٹ میں محفوظ کیا ہے۔

فنکشن (`input`) میں ہم نے یوزر سے ان پٹ لی ہے اور سٹرکچر کے ویری اینڈ لار استعمال کئے ہیں۔ اب سٹرکچر کے ویری اینڈ لار استعمال کرنے کے لئے ان کو ایک ریفرنس کی ضرورت چاہیے اس کے لئے ہم نے اس فنکشن میں `temp` پوائنٹر او بجیکٹ بنایا ہے اور یہ پوائنٹر ہے۔ اس لئے ویری اینڈ لار کو پوائنٹر ایس آپریٹر (\rightarrow) کی مدد سے چھینیں کیا ہے اور آخر میں پوائنٹر ریٹن کیا ہے یعنی پوائنٹر کا او بجیکٹ `temp` ریٹن کیا ہے۔ فنکشن میں ہم نے پوائنٹر او بجیکٹ بطور آر گومنٹس پاس کیا ہے اور اس کو مزید پراسنگ کے لئے استعمال کیا ہے۔

سٹرکچرز: Nested

آپ سٹرکچر میں مزید سٹرکچر بھی لکھ سکتے ہیں اور یہ کس طرح کام کرے گا اور اس کو لکھنے کا کیا طریقہ ہے آئیے اس کے لئے ایک پروگرام لکھتے ہیں۔

شانہ 5.4

```

struct premier
{
    int pm;
    float cm;
};

struct inner
{
    premier height;
    premier width;
};

void main(void)
{
    clrscr();
    inner inn; //instance of inner
    float templ, temp2;
    inn.height.cm=5.16; //assign values to inner
}

```

```
inn.width.pm=11;  
cout <<"\n Enter length of ground in inches:";  
cin >>inn.height.cm;  
cout <<"\n Enter width of ground in feet:";  
cin >>inn.height.pm;  
temp1=inn.height.pm+inn.height.cm/12;  
temp2=inn.width.pm+inn.width.cm/12;  
cout <<"\n Ground area is:" <<setprecision(2) <<(temp1*temp2);  
cout <<"squarefeet";  
getch( );  
}
```

اس پروگرام میں ہم نے دو سڑک ہر زبانے ہیں اور ایک سڑک پر کافی نہیں ویری اینٹل کیا ہے اور اس ایک ویری اینٹل کو مختلف سڑک پر کی ویلو آسانی کی ہیں۔ اس پروگرام میں پہلے premier نام کا سڑک پر بنا یا ہے اور دوسرا سڑک پر inner ہے۔ اس inner سڑک پر میں دو ویری اینٹل ہیں جن کی تائپ premier (یعنی پہلا سڑک) ہے۔ اس پروگرام کو جب آپ ایگر کیوں کریں گے تو یہ چار ویلوز کو کیلکو لیٹ کرے گا کیونکہ آپ کے پاس چار ویری اینٹل ہیں دو پہلے سڑک پر کے اور دو دوسرے سڑک پر میں ڈیلکٹر کے گئے ہیں۔ ایک لمحہ اس کہ آپ () main میں اس سڑک پر کا اوپیجیٹ یا اسٹینشنس ڈیلکٹر کریں گے۔ جس میں پہلے سڑک پر تائپ کے ویری اینٹل ڈیلکٹر ہوں گے۔ اس پروگرام کی کافی بہت یہ ہو گی۔

Enter length of ground in feet: 14
Enter length of ground in inches: 16.21
Ground area is: 178.22 square feet

لیورڈ یونیورسٹی میاں پس:

آپ نے C++ کی ڈیناٹا پس (int, char) کے بارے میں پڑھا ہے۔ C++ ان ٹائپس کے علاوہ یہ بھی سہولت فراہم کرتی ہے لے کر آپ اپنی خود سے پیش ڈیناٹا پس بھی بناتے ہیں۔ آپ ایسا کئی طریقوں سے کر سکتے ہیں۔ لیکن ہم یہاں پر ایک سادہ یوزر ڈیفائل ڈیناٹا پس کے بارے میں بتائیں گے۔ آپ enumeration کی مدد سے خود اک integer ڈیناٹا پس بناتے ہیں۔ اس کا جزء طریقہ ہے۔

```
enum typename{enumerator list};
```

یہاں پر شارت میں جو enum لکھا ہے C++ کی ورد ہے۔ typename سے مراد کوئی بھی ایڈنٹیفارز ہے جو یہ ظاہر کرتا ہے کہ یہ ابھی ڈیفائن کر گئے کاتا نام ہے اور اسٹ میڈیا فارم کے لئے کانسٹنٹ ایڈنٹیفارز کے نام ہوں گے۔ مثل

enum Sex (Male, Female);

اس سے آپ کا کوڈ آسان ہو جاتا ہے لیکن ایک بات کا خیال رکھیں کہ enum کا بہت زیادہ استعمال نہ کریں۔ وہ اس لئے کہ enumlist میں

visit our website: www.papersbook.Blogspot.com

کئے ہوئے ایڈنیفارڈ درست ہونے ضروری ہیں۔ اس طرح کے ایڈنیفارڈ آپ ڈیکلیر نہیں کر سکتے۔

```
enum Declare{c-, a+b, A, D}; //Error
```

آئیے اس کے لئے ایک سادہ پروگرام لکھتے ہیں تاکہ آپ کو اس کے استعمال کرنے کا بہتر طریقہ آجائے۔

مثال نمبر 5.5

```
//user defined data types
```

```
enum Days{mon, tues, thur, wed, fri, sat, sun};
```

```
void main(void)
```

```
{
```

```
clrscr( );
```

```
Days day1, day2;
```

```
day1 = wed;
```

```
day2 = sun;
```

```
int temp;
```

```
if(day1<day2)
```

```
{
```

```
temp = day2-day1;
```

```
cout << "\n Day(s) Between=" << temp;
```

```
cout << "\n Day1 comes before day2";
```

```
}
```

```
else
```

```
{
```

```
temp = day1-day2;
```

```
cout << "\n Days between=" << temp;
```

```
cout << "\n day1 comes after day2";
```

```
}
```

```
getch( );
```

```
}
```

اس پروگرام میں ہم نے enum ڈیکلیریشن میں وہ تمام نام لکھے ہیں جو کہ اس ناٹسپ کی ویلوز ہوں گی یعنی آپ اس ناٹسپ کے ویری استبلر کو یہ ویلوز آسانی کریں گے۔ ان ویلوز کو اسٹریٹر کہتے ہیں۔ جب آپ ایک دفعہ enum ناٹسپ ڈیکلیر کر لیتے ہیں تو اس کے بعد آپ اس ناٹسپ کے ویری استبلر بنانے کے اہل ہو جاتے ہیں۔

```
Days day1, day2;
```

آئیے اس کے لئے ایک اور یوں ناپ کا پروگرام لکھتے ہیں۔

//word count 5.6

```

enum phrase{No, Yes};           //No=0, Yes=1
void main(void)
{
    clrscr();
    phrase ph;
    ph = No;
    char ch;
    int count = 0;
    cout <<"Enter a phrase:";

    do {
        ch = getche();           //read, get character
        if(ch==' '|| ch=='\r')   //if space
        {
            if(ph==Yes)         //if word
            {
                count++;        //count word
                ph = No;          //for again count reset
            }
        }
        else                     //a normal word
        {
            if(ph==No)          //start of word
            ph=Yes;              set flag
        }
    while(ch!='\r');           //end of Enter Key
    cout <<"\n word(s) count are:" <<count;
    getch();
}

```

جب آپ اس پروگرام کو جائز کیوں کریں گے تو آؤٹ پٹ کچھ اس طرح ڈیلے ہوگی۔

Enter a phrase: I Love Islam

visit our website: www.papersbook.Blogspot.com

word(s) count are: 3

نوت: آپ یہ جملہ بتنا چاہیں مرضی لکھ لیں لیکن صرف ایک لائن پر لکھ سکتے ہیں۔ جو نئی Enter پر لیں کریں گے آپ کا جملہ ختم ہو جائے گا۔

اس پروگرام میں ہم نے ایک phrase نام سے ڈیکلیر کی ہے جس کی دو ویلوں ہیں۔ Yes اور No اس میں Yes ویلو 1 اور No کی ویلو 0 ہو گی۔ اور شارت میں ph ویری اسی طرز کیا ہے۔ یہ phrase ٹاپ کا ویری اسیل ہے بعد میں یوزر سے do لوپ میں ان پٹ لی ہے۔ فناشن اس وقت تک ان پٹ دے گا جب تک کہ آپ Enter پر لیں نہیں کرتے۔ اس کے بعد a کندش استعمال کی ہے کہ اگر لکھی گئی ان پٹ میں پسیں ہے تو کلی جملہ بھی ہے تو ++count یعنی count میں ایک کا اضافہ کروے اور ph=no کرو دے تاکہ کر سارے لفظ کو پڑھ سکے اور اگر ایسا نہیں ہے تو جتنے لفظ لکھے گئے ہیں اس کی تعداد پر نت کر دے۔

(Classes):

آپ نے ابھی تک C++ پروگرامنگ کیونکے متعلق بنیادی پروگرامنگ پڑھی ہے لیکن اب آپ ہائی یول پروگرامنگ پڑھیں گے اور صحیح معنوں میں پروفیشنل پروگرامنگ کا اس سے شروع ہوئی ہے۔ اور آپ کا او بجیکٹ اور ینڈ پروگرامنگ کا کانسٹرکٹ بھی یہاں سے شروع ہوتا ہے۔ کلاس ایک ڈرائیور یہ ٹاپ ہے یعنی یہ کسی اور جزا یا عنصر سے حاصل کی جائے اور اس کے عناصر کی پھر اور ناتھس ہوتی ہیں۔ ایک کلاس ایک ارے کی مانند ہوتی ہے لیکن ارے کے عناصر کی ٹاپ ایک ہوتی ہے جبکہ کلاس کے عناصر کی ٹاپ مختلف بھی ہو سکتی ہے۔ اس کے علاوہ کلاس کے عناصر ویری اسیل کے علاوہ فناشن یا آپ پڑھ بھی ہو سکتے ہیں۔ ایسے پروگرام جو کلاس استعمال کرتے ہیں اسے او بجیکٹ اور ینڈ پروگرامنگ کہتے ہیں۔ اس میں پروگرام ایک او بجیکٹ ماؤں بناتا ہے۔ ایک او بجیکٹ خود مختار استٹی ہے جو اپنا ذیٹا سور کرتا ہے اور اس کے اپنے فناشن بھی ہوتے ہیں۔ او بجیکٹ کا ایک اہم کام یہ جانا بھی ہے کہ کس ایکشن کو کب اور کیسے پر فارم کرنا ہے۔

او بجیکٹ میں کچھ کا اعزز ہوتے ہیں جنہیں ذیٹا مبرز کہتے ہیں اور اس میں کچھ مبرز فناشن بھی ہوتے ہیں۔ مگر فناشن کو میکٹر بھی کہا جاتا ہے اور یہ اس وقت کا ل ہوتا ہے جب او بجیکٹ کو انسٹرکشن پڑھی جاتی ہے۔ اپنے پروگرام میں کلاس شامل کرنا او بجیکٹ اور ینڈ پروگرامنگ کی طرف پہلا قدم ہوتا ہے۔

کلاس ڈیکلیریشن:

آپ کسی بھی طرح کی کلاس یوں ڈیکلیر کرتے ہیں۔

```
class first{
public:
    void sum(void);
    void print(int, int);
private:
    int num, den;
};
```

کسی بھی کلاس کا آغاز C++ کی ورڈ کلاس سے ہو گا اور اس کے بعد پھر کلاس کا نام لکھا جاتا ہے اور اس کے بعد درمیانی بریکٹ لگائی جاتی ہے اور

کلاس کا اختتام پھر بریکٹ سے ہو گا جس کے بعد یہی کالن لکھنا ضروری ہوتا ہے۔ ہماری کلاس میں نشان (sum) اور (print) ممبر فنکشنز ہیں کیونکہ یہ کلاس کے ممبرز ہیں۔ اسی طرح ویری اسپلر den اور num کلاس کے ڈیٹا ممبرز ہیں۔ آپ ممبر فنکشنز کو میتھڈز بھی کہہ سکتے ہیں۔ اس کلاس میں ہم نے میتھڈ سے پہلے public کی ورڈ لکھا ہے جبکہ ڈیٹا ممبر سے پہلے private لکھا ہے۔ یہ ایکسیس کی ورڈز ہوتے ہیں یا انہیں ایکسیس ناپ بھی کہتے ہیں۔ ان دونوں میں فرق یہ ہے کہ public ممبر اس کلاس سے باہر بھی ایکسیس (استعمال) کے جاسکتے ہیں جبکہ private ممبر کو آپ اس کلاس میں استعمال کر سکتے ہیں اور جب آپ کسی بھی کلاس کے ممبر کو باہر ایکسیس کرنے کی سہولت فراہم نہیں کرتے تو اس میکنائجی کو انفارمیشن پاسینگ کہا جاتا ہے ان کے بارے میں تفصیل آپ آگے پڑھیں گے۔ فی الواقع صرف ان دونوں کا فرق ذہن میں رکھیں اور کلاسز کو ایک مثال سے سمجھنے کی کوشش کریں۔

//Simple Class 5.7

```
class first{                                //class declaration
    private:
        int temp;                           //class data
    public:
        void assign(int a)                //member function
        {
            temp = a;
        }
        void print( ) {                  //end of class
            cout <<"You entered:" <<temp;
        }
    };
void main(void){                            //defining objects
    first obj1, obj2;                      //calling member functions
    obj1.assign(201);
    obj2.assign(107);
    obj1.assign( );
    obj2.assign( );
    getch( );
}
```

اس مثال میں ایک کلاس میں دو میتھڈ تحریر کئے ہیں جبکہ ایک ڈیٹا ممبر ہے جس کا ایکسیس مودیفایائر private ہے اور (main) میں اس کلاس کے دو ایکٹس بنائے گئے ہیں۔ او بھیکٹ کا معمولی ساتھار آپ نے سڑک مریں پڑھا اس کی تفصیل آگے پڑھیں گے۔ دو ایکٹس بنانے کا یہ فائدہ ہے کہ آپ ایک نشان کو دو دفعہ مختلف طیویز کے ساتھ بآسانی کال کر سکتے ہیں۔

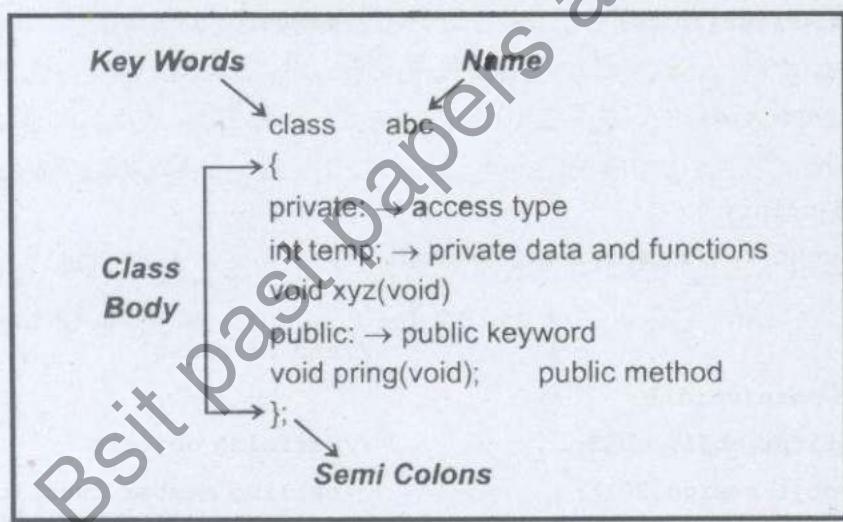
اویجیکٹس:

اویجیکٹ کا سب سے اہم جواب یہ ہے کہ اویجیکٹ کا کلاس سے وہی تعلق ہوتا ہے جو ایک دیری ایمیل کا اپنی ڈیٹا سپ سے ہوتا ہے۔ آپ ایک اویجیکٹ کا کلاس کو ایک انٹیس بھی کہہ سکتے ہیں۔

پرائیویٹ اور پبلک کی ورڈز:

آپ اکثر ہر ایک کے منہ سے ایک ہی لفظ سنتے ہیں کہ مجھے اویجیکٹ اور یونڈ پروگرامنگ کا کانپیٹ ہے۔ اس کو OOP بھی کہتے ہیں۔ OOP کا ایک اہم فچور ڈیٹا میٹنگ بھی ہے۔ اس کا یہ مطلب ہے کہ ڈیٹا کو ایک کلاس میں محدود کر دیا جائے اور کوئی دوسرا پروگرام اسے اس کلاس سے باہر نکال سکے۔ اس کے لئے اسے پہلے کی ورڈ private لکھا جاتا ہے۔ پرائیویٹ ڈیٹا یا میٹنگز صرف اسی کلاس میں ایکسیس کے جاسکتے ہیں جبکہ public ڈیٹا یا میٹنگز آپ ایک کلاس سے بھی کہیں بھی استعمال کر سکتے ہیں۔

ہم نے اوپر ممبر فنکشنز کا ذریعہ مدد یہے نکشہ جو کلاس کی باؤنڈی میں شامل ہوں یا جو کلاس کا ایک حصہ ہوں وہ ممبر فنکشنز کہلاتے ہیں اور اکثر ایک کلاس کے ممبر فنکشنز یا میٹنگز پبلک جبکہ ڈیٹا میٹنگز پرائیویٹ ڈیٹا کیلئے کے جاتے ہیں۔ ڈیٹا اسی لئے پرائیویٹ ڈیٹا کیلئے کیا جاتا ہے تاکہ یہ محفوظ رہ سکے اور پروگرام کا دوسرا حصہ اسے ڈسٹریب نہ کرے۔ آپ ایک کلاس کے کام کرنے کا طریقہ اس حആ رکی سے سمجھ سکتے ہیں۔



اب آپ نے کلاسز کے بارے میں کافی بنیادی معلومات حاصل کر لی ہیں۔ اس کے لئے اب ایک پروگرام لکھتے ہیں جو مزید آپ کی رہنمائی کرے گا۔

مثال نمبر 5.8

```
class employee
{
    private:
        char name[20], address[25];
        int salary;
```

```

public:
    int empno;
    void input(void);
    void output(void);
};

void employee::input(void)                                //function definition
{
    cout << "Employee Number";
    cin >> empno;
    cout << "Enter Name";
    gets(Name);
    cout << "Enter Address";
    gets(Address);
    cout << "Enter Salary";
    cin >> Salary;
}
//end of function

void employee::output(void)                             //function definition
{
    cout << "Name is:" << name;
    cout << "Address is:" << address;
    cout << "Salary is:" << salary;
}
//end of function

void main(void)
{
    clrscr();
    employee emp;                                     //object declaration
    emp.input();                                       //function calling
    emp.output();
    getch();
}

```

یہاں پر ہم نے `emp` کا اوبجیکٹ ذیکر کیا ہے۔ اس اوبجیکٹ کے اپنے ذیاً ممبرز اور میتھڈز میں اور یہ اوبجیکٹ اس لئے ذیکر کیا ہے تاکہ یہ اپنے میتھڈز (اوپر `input()` اور `output()`) کو کال کرے۔ آپ نے دیکھا کہ میتھڈ کال کرنے کے لئے اوبجیکٹ کا نام پھر ذات آپریٹر اور آخر پر میتھڈ

کا نام لکھا جاتا ہے۔ ایک او بجیکٹ بالکل ایسے ہی ڈیکلیر کیا جاتا ہے جس طرح کہ ایک وری دیبل ڈیکلیر کیا جاتا ہے لیکن فرق صرف یہ ہے کہ اس کی ڈینا ٹانپ کلاس ہوتی ہے اور آپ اسے اس طرح یوزر ذیفائنن ڈینا ٹانپ بھی کہہ سکتے ہیں۔

آپ نے دیکھا کہ جب کلاس کے میخفز کلاس سے باہر ذیفائنن کے جاتے ہیں تو ان کو لکھنے کا شاکل قدرے مختلف ہونا چاہئے۔ وہ اس لئے کہ یہ کلاس کے نمبر ہوتے ہیں اور انہیں اس کلاس کے ریفس سے باہر ذیفائنن کیا جاتا ہے۔ میخفز کی ذیفائنن کے لئے یہ لکھنا ضروری ہے۔ اس میں پہلے کلاس کا نام اور اس کے بعد ڈبل کالن دو دفعہ لکھا جاتا ہے۔

```
employee::input(void)
```

ان کا نز کو جزویشن آپ سٹر (:) یا سکوپ ریز لوشن آپ بیڑ بھی کہتے ہیں۔

آپ نے اپنے مثال بفر 5.8 میں ایک کلاس بنائی ہے اور اس میں آپ ایک دفعہ یوزر سے ان پٹ لے رہے ہیں جنی آپ صرف ایک ملازم کا ڈینا حاصل کر رہے ہیں اور وہ نہ کر رہے ہیں۔ اگر آپ ایک سے زیادہ ملازم میں کا ڈینا حاصل کرنا چاہتے ہیں تو اس کے لئے آپ کو یہ فنشن مطلوبہ نامنز میں کال کرنا ہو گا۔ جیسا کہ ہم میثال بفر 5.7 میں دو دفعہ ایک فنشن کال کیا تھا۔ اس مثال میں ہم نے دو او بجیکٹس بنائے تھے لیکن اگر آپ دس ملازم میں کا ریکارڈ حاصل کرنا چاہتے ہیں تو کیا دس او بجیکٹس ڈیکلیر کریں گے تو اس کا جواب یہ ہے کہ نہیں تو پھر ایسا کیسے ہو گا؟ اس کے لئے آپ او بجیکٹ ارے ٹانپ کو ڈیکلیر کریں گے۔ آئیے اس کا حل دیجئے۔ اس کے لئے آپ کا باتی کوڈ 5.8.1 main کا () main میخفز دوبارہ لکھ رہے ہیں۔

5.8.1 حل نمبر

```
//Rest of code 5.8
void main(void)
{
    clrscr();
    employee emp[4];
    int count=0;
    for (; cout<<4; cout++)
    {
        cout << endl;
        emp[count].input();
    }
    for( ; count<4; count++)
    {
        cout << endl;
        emp[count].output();
    }
    getch();
}
```

اس پروگرام میں دو فنکشن یا میٹھڈز چار دفعہ کال ہو رہے ہیں کیونکہ آپ نے کلاس کا او بجیکٹ ایک ارے بنایا ہے اور اس کو چار ویلیوز پاس کی ہیں اور لوپ کو بھی ہم نے چار دفعہ ایگزیکوٹ کیا ہے۔ اب جب آپ اسے ایگزیکوٹ کریں گے تو یہ چار ملازم میں کاریکارڈ پوچھے گا اور چار ملازم میں کے بارے میں معلومات ڈپلے کرے گا۔

کنسٹرکٹر:

جب ایک کلاس بنائی جاتی ہے تو اس کے ممبر کلاس کے کنسٹرکٹرن کی مدد سے ائمی ٹھلاڑ کے جاسکتے ہیں۔ ایک کنسٹرکٹر کلاس کا ممبر فنکشن ہوتا ہے اور اس کا نام اور کلاس کا نام ایک ہی ہوتا ہے اور پروگرام جو کنسٹرکٹر فراہم کرتا ہے وہ جب بھی کلاس انسٹینس (instance) کا بناتا ہے خود بخود کال کر لیا جاتا ہے۔ آپ کنسٹرکٹر کو اور لوپ کی بھی مدد کر سکتے ہیں۔ اس کا کیا طریقہ ہے آپ آگے پڑھیں گے۔ ذیباً ممبرز کو کلاس انسٹرکٹر میں بھی ٹھلاڑ کرنا چاہئے یا ان کی ویلیوز اس وقت سیٹ کی جاتی ہے جب کلاس کا او بجیکٹ بنایا جاتا ہے۔ آئیے کنسٹرکٹر کے لئے ایک پروگرام بناتے ہیں۔

مثال نمبر 5.9 کلاس انسٹرکٹر

```

class base
{
public:
    base(int a, int b); //constructor
    {
        num=a;
        den=b;
    }
private:
    int num, den;
    void sum();
};

void main(void)
{
    clrscr();
    base c(5, 2), f(7, 13); //object
    cout << "First call c=";
    c.sum();
    cout << "\n Second call f=";
    f.sum();
    getch();
}

```

```

void base::sum( )
{
    num+=den;
    cout <<num << "+" <<den <<" = " <<num;
}

```

اس مثال میں ہم نے base نام کی ایک کلاس بنائی ہے اور اس میں ایک (base) main کا فنکشن بھی ہے۔ یہ اصل میں کلاس کا کنٹرولر ہے اور یہ ڈیٹا ممبرز کو مخصوص ویلیوں آسانی کرتا ہے جب (main) میں موجود ڈیلکٹریشن ایگزیکوٹ ہو گی تو کنٹرولر خود بخود کال کر لیا جائے گا اور ویلیو 5 اور 2 اس میں موجود پیر ایمپلز میں سے کلاس کر دی جائے گی۔ پھر فنکشن (base) یہ ویلیو num اور den کو آسانی کر دے گا۔ اصل میں ایک کلاس کا کنٹرولر کلاس کے او بجیکٹ کے لئے اس میں پسیں ریزو کرتا ہے آپ ایک کلاس کے ایک سے زائد کنٹرولر بھی بناتے ہیں یہ آپ آگے پڑھیں گے۔ آئیے اس وقت ہم ایک اور مثال لکھتے ہیں۔

مثال نمبر 5.10 کنٹرولر

```

class temp
{
private:
    usnsigned int watch;
public:
    temp( )
    {
        watch = 12;
    }
};

void main(void)
{
    clrscr( );
    temp t;
    cout <<"t.watch() = " <<t.watch();
    getch( );
}

```

بعض کنٹرولر زصرف ڈیٹا کو اپنی ہلائز کرنے کے لئے ہی استعمال ہوتے ہیں۔ اس کے لئے C++ ایک الگ سے طریقہ فراہم کرتا ہے جسے اپنی شلائریشن سلت کہتے ہیں۔ اس کے لئے آپ یہ کوڈ پڑھیں گے۔

```
base(int a, int b): num(a), den(b){ }
```

اینی شاپریزیشن لست (:) کالن سے شروع ہوتی ہے اور فناشن باؤڈی پر ختم ہوتی ہے اور فناشن باؤڈی اس وقت خالی ہے۔

کنسٹرکٹر اور لوڈنگ:

ہم نے اس باب میں پہلے بھی ذکر کیا تھا کہ آپ ایک کلاس کے ایک سے زائد کنسٹرکٹر بھی بنائے ہیں اس کو کنسٹرکٹر اور لوڈنگ کہتے ہیں۔ یہ بالکل فناشن اور لوڈنگ کی طرح کام کرتا ہے اور C++ کپا مکر کنسٹرکٹر کے آر گو میٹ سے شناخت کرتا ہے کہ اب کون سافناشن کال کیا جا رہا ہے۔ آئیے اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 5.11 کنسٹرکٹر اور لوڈنگ

```
class Base
{
    private:
        int xcord;
        float inches;
    public:
        Base( ): xcord(0), inches(0.0)
        { } //initialization line
        Base(int xc, float in)
        {
            xcord=xc;
            inches=in;
        }
        Base(int xc): inches(6.4)
        {
            xcord=xc;
        }
        void getdata()
        {
            cout << "\n Enter x-coordinate of Rectangle in integer:" ;
            cin >> xcord;
            cout << "\n Enter length in inches:" ;
            cin >> inches;
        }
        void display( )
        {
            double ans;
```

```

ans=(xcord+inches)/2;

cout <<"\n Answer=" <<ans;

}

};

void main( );
{
clrscr( );

Base b1(13), b2(10, 5.6), b3;
b3.getdata();
cout <<"\n b1(13):";
b1.display();
cout <<"\n b2(10, 5.6):";
b2.display();
cout <<"\n b3.getdata():";
b3.display();
getch();
}

```

اس پروگرام میں ہم نے ایک ڈیفائل کنسٹرکٹر بھی لکھا ہے اس میں کوئی آر لوٹ پاس نہیں کیا گیا یعنی (base کنسٹرکٹر ہماری کلاس کا ڈیفائل کنسٹرکٹر ہے اور جب بھی کلاس کا اوجیکٹ بنے گا یہ خود بخوبی کال ہو جائے گا اس کے علاوہ اس کا ہم اسی میں نے ایسی ٹلائزیشن لائے بھی استعمال کی ہے اس میں ہم نے تین کنسٹرکٹر لکھے ہیں۔ ایک کو کوئی پیرامیٹر پاس نہیں کئے دوسرا میں base(int xc) صرف ایک پیرامیٹر پاس کیا ہے جبکہ تیسرا کنسٹرکٹر (base(int xc, float in) کو دو پیرامیٹر پاس کئے ہیں اور کپاٹر ان پیرامیٹرز کی مدد سے یہ کرتا ہے کہ اب کون سا کنسٹرکٹر کال کیا جا رہا ہے۔ جب آپ اس پروگرام کو ایگزیکیوٹ کریں گے تو یہ آٹوٹ پٹ ڈسپلے ہو گی۔

Enter x coordinate of rectangle in integer: 12

Enter length in inches: 8.24

b1(13):

Answer = 9.7

b2(10, 5.6):

Answer = 7.8

b3.getdata():

Answer = 9.12

اویجیکٹ بطور آر گومنٹس:

آپ فنشن پیرا میٹر ز اور آر گومنٹس سے بخوبی واقف ہیں اور آپ نے اویجیکٹ بنانا بھی سیکھا ہے۔ ہماری اگلی مثال کچھ مختلف ہے اس میں ہم ایک فنشن کو کلاس کا اویجیکٹ پاس کریں گے۔ یہ کس طرح کام کرے گا آئیے اس کو سمجھتے ہیں۔

مثال نمبر 5.12 اویجیکٹ بطور فنشن آر گومنٹس

```

class Base
{
private:
    int xcord;
    float length;
public:
    Base( ): xcord(0), length(0.0)
    {
    }
    Base(int xc): inches(6.14)
    {
        xcord=xc;
    }
    void getdata( )
    {
        cout <<"\n Enter x-coordinate in integer:" ;
        cin >>xcord;
        cout <<"\n Enter length in inches(i.e 3.7):" ;
        cin >>length;
    }
    void display( )
    {
        cout <<"x-coordinate: " <<xcord <<"\n length: " <<length;
        double calculate(Base, Base);
    };
    double base::calculate(Base b1, Base b3);
    {
        double ans;
        length = b1.length+b3.length;      //getvalues of object b1&b3
    }
}

```

```

record = b1.xcord+b3.xcord;
ans = (xcord+length)/2;
return ans; //return answer
}
void main(void);
{
clrscr();
Base b1(13), b3;
double temp;
b3.getdata();
cout <<"\n With default value b1(13)\n";
b1.display();
temp = b3.calculate(b1, b3); //value receiving
cout <<"\n b3.display()\n";
b3.display(); //after assinging new values in calculate()
cout <<"\n Answer of calculate() ; " <<temp;
getch();
}

```

آپ نے دیکھا کہ ہم نے اس پروگرام میں double calculate(Base, Base) فنکشن بنایا ہے۔ اس کو بعد میں Base ٹائپ کے "اویجیٹ بھی پاس کئے گئے ہیں اور اس فنکشن کی باؤسی میں ہم نے کلاس کے ڈیٹا میمبرز کو دوالیں تیکش کی مدد سے ایکسیس کیا ہے اور ان کا مجموعہ معلوم کیا ہے اور بعد میں اس کو 2 سے تقسیم کیا گیا ہے اور ان کا جواب ریٹرن کیا ہے اور یہاں سے جو ویور پڑتی ہو رہی ہے اسے بطور() main میکھڑ میں دیری اسٹبل میں محفوظ کیا ہے۔ جب آپ اس پروگرام کو انگریزی میکیوٹ کریں گے تو یہ آٹ پت ڈپٹے ہو گے۔

```

Enter x-coordinate in integer : 17
Enter length in inches (i.e 3.7) : 5.2
with default value b1(13)
x-coordinate : 13
length : 6.14
b3.display()
x-coordinate : 30
Answer of calculate() : 20.67

```

فناشن سے او بجیکٹ ریٹرن کرنا:

آپ نے اس سے پہلے ایک پروگرام لکھا تھا جس میں او بجیکٹ بطور آر گومٹ پاس کیا تھا۔ اسی طرح آپ ایک فناشن کی ریٹرن نائپ کسی ڈپٹ نائپ کی بجائے او بجیکٹ بھی واضح کر سکتے ہیں۔ لہن آپ فناشن ایک او بجیکٹ ریٹرن کرے گا۔ یہ کس طرح ہو گا آئیے اس کے لئے ایک پروگرام لکھتے ہیں۔

••••• نمبر 5.13 فناشن سے او بجیکٹ ریٹرن کرنا

```

class Super
{
    private:
        int num, power, fact;
    public:
        Super(): fact(1)
    {
    }
    void getdata()
    {
        cout << "Enter a positive number:" ;
        cin >> num;
        cout << "\n Enter its power:" ;
        cin >> power;
    }
    void display()
    {
        cout << "\n Answer: " << fact;
    }
    Super calculate(Super super);
};

Super Super::calculate(Super ans)
{
    //super ans;
    ans.fact=pow(num, power);
    return ans;
}

```

```
{
clrscr( );
Super s1, temp;
s1.getdata();
temp=s1.calculate(temp);
temp.display();
getch();
}
```

اس پروگرام میں ہم نے ایک فنکشن Super calculate(Super) بنایا ہے اس فنکشن کی ریٹرن ٹائپ کلاس کا اوجیکٹ ریٹرن کریں گے۔ یہ اب ہم نے جب فنکشن کی باڑی یا ذہنیت کھی ہے تو بریکس میں Super Super::calculate(Super ans) ڈیکلائر کیا تھا۔ اگر آپ فنکشن کو لوں آدمیت پاس نہیں کرنا چاہتے تو (Super calculate() کو ایسے لکھیں اور اس کی باڑی میں کلاس کا اوجیکٹ بنالیں جیسا کہ ہم نے کوئی میں میں بنایا ہے۔

//Super ans;
بعد میں ہم نے اس اوجیکٹ کو ریٹرن کیا ہے; main() اور returnans میں میکس میں اس ریٹرن کی ہوئی ویڈیو کو ایک اوجیکٹ میں محفوظ کیا ہے۔

```
temp=s1.calculate(temp);
```

اب ہم اس حاصل کی گئی ویڈیو کو ریٹرن کرنا چاہتے ہیں۔ یہی، ہم نے فنکشن calculate میں جو آپ نے پر فارم کیا ہے اسے ڈسپلے کرنا چاہتے ہیں تو اس کے لئے اس اوجیکٹ temp کے ریفرنس سے (display) فنکشن کا لائی ہے۔ ہمارا یہ پروگرام یوزر سے دو نمبر بطور ان پٹ لے گا اس میں دوسرا نمبر پہلے نمبر کی پاور ہو گا یعنی وہ اتنی دفعہ پہلے نمبر کو آپ میں ضرب دے گا اس کے لئے ہم نے (pow) فنکشن استعمال کیا ہے۔

```
pow(num, power);
```

فنکشن C++ کی pow() ہڈی رفائل میں ڈیفائن کیا گیا ہے۔

(Destructor):

ایک ڈسٹرکٹر ایک کلاس کا ایک پیشہ میں فنکشن ہوتا ہے۔ آپ نے دیکھا ہے کہ جب ایک اوجیکٹ بلا ہجت ایک ڈسٹرکٹر اوجیکٹ کو منظوم کرنے کے لئے خود بخوبی بن جاتا ہے۔ اسی طرح جب ایک اوجیکٹ ختم ہوتا ہے تو ایک اور پیشہ میں فنکشن کا لیو ہوتا ہے جو اس اوجیکٹ کے ختم ہونے کو منظم کرتا ہے اس خصوصی فنکشن کو ڈسٹرکٹر کہتے ہیں جس کی کلاس کے لئے ڈسٹرکٹر آپ خود بھی لکھ سکتے ہیں۔ اس میں کلاس کے نام سے پہلے میلہ (~) علامت لکھی جاتی ہے۔ یہ علامت اصل میں پلیسیٹ کے لئے استعمال ہوتی ہے اور ڈسٹرکٹر کلاس کے ڈسٹرکٹر کا پلیسیٹ ہے اس لئے اس کے ساتھ یہ علامت لکھی جاتی ہے۔

ہر کلاس کا صرف ایک ڈسٹرکٹر لکھا جاتا ہے اور اگر یہ خود ڈیفائن نہ کیا جائے تو ڈسٹرکٹر کی طرح یہ خود بخوبی کا لیو ہوتا ہے۔ ڈسٹرکٹر کی طرح اس کی کوئی ریٹرن ٹائپ نہیں ہوتی اور نہ ہی اس کو کوئی پیرامیٹر پاس کیا جاتا ہے۔ یہ اصل میں کلاس کے فالتو ڈیباگرز سے میوری آزاد کرواتا ہے یعنی جب کلاس کا اوجیکٹ ختم ہو جاتا ہے تو پھر اس کے ڈیباگرز میں موجود ویڈیو ہمارے کسی کام کی نہیں ہوتی۔ تو ان کو ختم کر دینا چاہتے تاکہ ہماری میوری مزید کسی کام کے لئے استعمال کی جاسکے یہ کام ڈسٹرکٹر کرتا ہے۔ آئیے اس کی ایک آسان سی مثال دیکھتے ہیں۔

مثال نمبر 5.14 ڈسٹرکٹر اور کنٹرکٹر کا لئے

```

class Destructor
{
private:
    int temp;
public:
    Destructor( ): temp(4)           //constructor
    {
        cout <<"\n\n constructor called:" <<temp;
    }
    ~Destructor( )                  //Destructor
    {
        temp = 0;
        cout <<"\n\n Destructor called:" <<temp;
    }
};                                //end of class

void main(void)
{
    clrscr();
    {                               //scope of Object
        Destructor d;
        cout <<"\n now object is born:";

        //scope ended
        cout <<"\n outside the range of object:";

    }                               //againg Object scope
    Destructor a;
    cout <<"again object is born:";

}
getch();
}

```

اس پروگرام کو جب آپ ایگز کیوٹ کریں گے تو اس پروگرام کی یہ آٹھ پت ہوگی۔

constructor cailed : 4

visit our website: www.pupapersbook.blogspot.com

now object is born :

Destructor called : 0

outside the range of object :

Constructor called : 4

again object is born :

destructor called : 0

اس پر دلیل اسی ہے کہ جو نئی اوبجکٹ کا سکوپ ختم ہوتا ہے تو ذمہ کرنا کال ہوتا ہے اور اس کا مذہبیت میں ایگز کیوٹ ہوتی ہے۔

مشق

سوال نمبر 1: ستر کچر کے مبڑ کو ایکسیس کیسے کیا جاتا ہے؟

سوال نمبر 2: ستر کچر استعمال کرتے ہوئے ایک پروگرام تحریر کریں جو یوزر سے دو پاؤں کی ویلوں لے۔ مثلاً

Enter coordinates for point1 = 3 4

Enter coordinates for point2 = 8 3

پھر آپ کا پروگرام تیراپاؤں کو معلوم کر کے اور یہ پاؤں کو ان دونوں coordinates کو جمع کرنے سے حاصل ہو گا۔

سوال نمبر 3: ان سوالات کا مختصر جواب تحریر کریں؟

(i) C++ میں کلاس اور ستر کٹر میں کیا فرق ہے؟

(ii) کلاس کٹر کٹر اور ڈسٹر کٹر میں کیا فرق ہے؟

(iii) ایک کلاس کے پیلک مبڑ اور پرائیویٹ مبڑ میں کیا فرق ہے؟

(iv) کلاس ڈیفینیشن میں سکوپ ریزولوشن آپریٹر (::) کیوں اور کچھ استعمال کیا جاتا ہے؟

(v) کلاس کے کٹر کٹر اور ڈسٹر کٹر کا کیا نام ہونا چاہیے؟

سوال نمبر 4: پروگرام کے ہر حصے میں کیا ایرہ ہے اور آپ اسے کیسے ختم کریں گے؟

(i) class abc

public:

int abc(const int*, int);

private:

//private member

}

فرض کریں کہ یہ کوڈ ایک کلاس abc میں لکھا ہے۔ (ii)

void ~abc(int, char);

(iii) *a.temp;

(iv) class abc

{

```

        public:
            //member function
        private:
            int temp = 0;
            int count = 0;
        };

class xyz
{
public:
    xyz(int a, xyz* x=0)
    {
        data = a;
        count = x;
    }
    int data;
    xyz* count;
};

void main( )
{
    clrscr( );
    int f;
    xyz* x;
    xyz* y;
    while(cin>>f)
    {
        x=new xyz(f, y);
        y=x;
    }
    for(;x->count; x=x->count;
        cout <<x->data<<"->";
        cout <<"\n";
}

```

سوال نمبر 5: اس پروگرام کی آٹھ پٹ کیا ہوگی؟

جوابات

1: جواب

جواب: سڑکھر کے مہر زکو ایکسیں کرنے کے لئے دو ایکسیں آپ پریز، ڈاٹ آپ پریز (.) اور آر آپ پریز (→) استعمال کئے جاتے ہیں۔ ڈاٹ آپ پریز کی مدد سے آپ سڑکھر کے او بھیکٹ کار بیزنس ایکسیں کرتے ہیں جبکہ آر آپ پریز اس وقت استعمال کرتے ہیں جب آپ او بھیکٹ کو پوائنٹ ڈیکلیر کرتے ہیں۔ آپ او بھیکٹ پوائنٹ ڈیکلیر کرتے ہیں۔ آپ اس طرح ان آپ پریز کو استعمال کرتے ہیں۔

Question q2;

```
Question *qptr;
cout << q2.temp;
cout << qptr->temp;
```

2: جواب

```
struct Question2
{
    int xco;
    int yco;
};

void main(void)
{
    clrscr();
    Question2 q1, q2, q3;
    cout << "\n Enter coordinates for point1:" ;
    cin >> q1.xco >> q1.yco;
    cout << "\n Enter coordinates for point2:" ;
    cin >> q2.xco >> q2.yco;
    q3.xco=q1.xco+q2.xco;
    q3.yco=q1.yco+q2.yco;
    cout << "\n coordinates of q1+q2=" << q3.xco << ", " << q3.yco;
    getch();
}
```

3: جواب

- (i) C++ میں کلاس اور سترکٹر تقریباً ایک ہی ہیں۔ صرف ایک نمایاں فرق ہے کہ کلاس میں ڈینالٹ ایکسیس لیول پر انجیویٹ ہوتا ہے جبکہ سترکٹر پلک ہوتا ہے۔
- (ii) ایک لنسٹر کلاس کا ممبر فنکشن ہوتا ہے اور یہ اس وقت خود بخواہیگز کیوٹ ہوتا ہے جب اس کلاس کا او بجیکٹ بنایا جاتا ہے اور یہ دی اسٹبلز کو اپنی شلائر کرتا ہے جبکہ ڈسٹرکٹر کلاس کا ایسا ممبر فنکشن ہے جو اس وقت خود بخواہیگز کیوٹ ہوتا ہے جب کلاس کے او بجیکٹ کا سکوپ ختم ہوتا ہے۔
- (iii) پلک ممبر کلاس سے باہر بھی ایکسیس کیے جاسکتے ہیں جبکہ پر انجیویٹ ممبر کلاس سے باہر ایکسیس نہیں کیے جاسکتے۔
- (iv) سکوپ آپریٹر: کسی بھی ریفرنس کے لئے استعمال کیا جاتا ہے۔ یہ کلاس کی ڈیفینیشن کلاس کے سکوپ سے باہر لکھتے ہیں۔
- (v) ہر کلاس کے کامن کا لکل وہی نام ہونا چاہئے جو کلاس کا ہو جبکہ ڈسٹرکٹر کا بھی وہی نام ہونا چاہئے جو کلاس کا ہے صرف اس کے شروع میں آپریٹر (~) نہیں بلکہ (Hide) لکھا جاتا ہے۔

4: جواب

- (i) لنسٹر کی زو بیوریزن نہیں کر سکتے اس کو درست کرنے کے لئے اس کے شروع میں ریزن نائپ int ختم کر دیں۔
- (ii) ڈسٹرکٹر کسی بھی قسم کی ویلیوریزن نہیں کرتا لورن یہ کسی قسم کے آر گومنٹ پاس کرتا ہے۔ اس کو درست کرنے کے لئے ریزن نائپ اور پیر ایمٹ ختم کر دیں۔
- (iii) آپ اس طرح ڈائریکٹ ایکسیس نہیں کر سکتے کیونکہ اس پر آپریٹر (.) کا priority لیول مشریک (*) سے زیادہ ہے اس کے لئے آپ کو اس کے گرد بریکٹس لگانی ہوگی۔

(*a).temp

- (iv) ممبر کلاس ڈیفینیشن میں ایکسیس نہیں کے جاسکتے۔ اس لئے یہاں پر صرف ان کو ڈیلکٹر کریں۔

5: جواب

اس پروگرام کی آڈٹ پٹ یہ ہوگی۔

75 71 33 12 7 ^D

71 → 12 → 33 → 71 → 75 →

باب نمبر 6

آپریٹر اور لوڈنگ

اس باب میں آپ C++ کے آپریٹر کو کام اور جیکٹ کے ساتھ استعمال کریں گے اور اس پر ایس کو آپریٹر اور لوڈنگ کہتے ہیں اور یہ C++ کی دوسری لینکو ہجر کی نسبت اضافہ فیچر ہے۔ لیکن آپریٹر اور لوڈ کرتے وقت ایک بات کا خاص خیال رکھیں کیونکہ اس کا غلط استعمال آپ کے پروگرام کو بہت مشکل بنادے گا۔ جیسا کہ >> آپ C++ میں کئی آپریٹر پر فارم کرنے کے لئے استعمال ہوتا ہے اور آپ اس کو اور لوڈ بھی کر سکتے ہیں لیکن اس بات کا خیال رکھیں کہ آپ کس کام کے لئے اس اور اور لوڈ کر رہے ہیں۔ آپ جو بھی آپریٹر اور لوڈ کرتے ہیں پہلا آپ کے آپریٹر استعمال کرنے کے مطابق اس کے لئے کوڈ لکھتا ہے۔ آپریٹر اور لوڈنگ کی اہمیت کو دیکھتے ہوئے ہم نے اس باب میں مندرجہ ذیل عنوانات شاہک کئے ہیں۔

آپریٹر اور لوڈنگ

آپریٹر کی ورث

اور لوڈنگ باسٹری آپریٹر

ملنی پل اور لوڈنگ

مشق

فرینڈ فنکشن

فرینڈ کلاس

کی ورث This

ڈینامیکس static

فونکشن مبرز static

(Friend Function):**فرینڈ فنکشن:**

آپ نے اس کتاب کے باب نمبر 5 میں پڑھا ہو گا کہ کوئی بھی نام ممبر فنکشن اور جیکٹ کے پرائیویٹ ڈیٹا ممبر کو ایکسیس نہیں کر سکتا۔ یعنی کہ اگر آپ ممبر فنکشن نہیں تو آپ پرائیویٹ یا پر ڈیکلر (Private or Protected) (Private or Protected) ڈیٹا ممبر کو ایکسیس نہیں کر سکتے۔

فرینڈ فنکشن بھی ایک نام ممبر فنکشن ہوتا ہے۔ آپ سوچ رہے ہوں گے کہ پھر اس کا کیا فائدہ ہے۔ تو فرینڈ فنکشن ایسا نام ممبر فنکشن ہے جو کلاس میں ڈیکلر کے لئے تمام ڈیٹا ممبر کو ایکسیس کر سکتا ہے خود وہ پرائیویٹ ہوں یا پر ڈیکلر۔ تو یہ اس کا سب سے بڑا فائدہ ہے کہ ایک کلاس کا ممبر فنکشن نہ ہوتے ہوئے بھی یہ کلاس کے تمام ڈیٹا ممبر کو ایکسیس کر سکتا ہے۔ یہ زیادہ تر آپریٹر اور لوڈنگ میں استعمال ہوتا ہے۔ اس فنکشن کو کیسے بناتے ہیں آئیے اس کے لئے ایک پروگرام دیں۔

مثال نمبر 6.1 فرینڈ فنکشن

```
class abc; //need for friend function

class xyz {
    friend int function(xyz, abc); //friend function
private:
    int temp;
public:
    xyz(int a)
    {
        temp = a;
    }
};

class abc
{
    friend int function(xyz, abc);
private:
    int temp;
public:
    abc(int a)
    {
        temp = a;
    }
};

int function(xyz x, abc a)
```

```

    {
        return(x.temp* a.temp);
    }

void main(void)
{
    clrscr();
    xyz x(4);
    abc a(7);
    cout << "\n Answer x(4), a(7) = " << function (x, a);
    getch();
}

```

اس پروگرام میں دو کلاسز xyz اور abc ہیں اسی تین ہم نے سب سے اوپر class abc ڈیکلائر کی ہے وہ اس لئے کہ اس کو فرینڈ فنکشن میں استعمال کیا گیا ہے اور آپ دیکھیں گے کہ فرینڈ فنکشن کی ورڈنگ friend سے ڈیکلائر کیا گیا ہے اور یہ دونوں کلاسز کا ممبر فنکشن نہیں ہے۔ اس پروگرام میں دونوں کلاسز کے نسٹر کمز کو ایک آر گومنٹ پاس کیا گیا ہے اور فرینڈ فنکشن میں دونوں کلاسز کے پر ایجوبیٹ ڈیٹا ممبر (int temp;) کو استعمال کیا گیا ہے۔ اس پروگرام کی آٹھ پٹ کچھ یوں ہوگی۔

Answer x(4), a(7) = 28

ہم نے فرینڈ فنکشن کو دو چیرا میٹر پاس کئے ہیں جو کہ کلاس کے انجکسٹس میں ہیں اسی فنکشن سے پہلے لکھا گیا کی ورڈنگ friend ختم کریں اور پھر اس پروگرام کو چلانیں تو یہ ایردے گا۔

```

xyz::temp is not accessible
abc::temp is not accessible

```

فرینڈ کلاس:

اوپر آپ نے ایک فرینڈ فنکشن بنانا سیکھا۔ آپ اس کے علاوہ آپ ایک کلاس کے تمام ممبر فنکشن ایک ہی دفعہ فرینڈ فنکشن بناتے ہیں لیکن یہ اس وقت ممکن ہو گا جب آپ ایک کلاس کو فرینڈ ڈیکلائر کریں گے۔ یہ کس طرح ممکن ہے آئے دیکھتے ہیں۔

مثال نمبر 6.2 فرینڈ کلاس

```

class temp
{
friend class second;
private:
int a;
int function()
{

```

```

        return a*a;
    }

public:
    temp (int x)
    {
        a=x;
    }
};

class second
{
public:
    void sec1(temp t)
    {
        cout <<"\n private data member value=" <<t.a;
        cout <<"\n private member function value=" <<t.func();
    }
};

void main(void)
{
    clrscr();
    int x;
    cout <<"\nEnter parameter value of object:" ;
    cin >>x;
    temp tp(x);
    second sd;
    sd.sec1(tp);
    getch();
}

```

اس پروگرام میں ہم نے دو کلاسز temp اور second بنائی ہیں اور second کا اس کو فرینڈ ڈیکلائر کیا ہے۔ اب یہ دوسرا کلاس second پہلی کلاس کے تمام ڈیتا میبز کو ایکسیس کر سکتی ہے۔ ہم نے temp کلاس کا نسٹر کنٹر لکھا ہے جس کو ایک پیر ایمیٹر پاس کیا گیا ہے اور اس کی ویبوہ ہم یوزر سے لے رہے ہیں۔ کلاس second میں sec1() کا ایک فنکشن لکھا ہے جس کو کلاس temp کا او بجیکٹ پاس کیا ہے اور اس کی مدد سے temp کلاس کے تمام پرائیویٹ ڈیتا میبز ایکسیس کئے جائیں۔ اس پروگرام کی آٹھ پٹ یہ ہوگی۔

Enter parameter value of object : 8

private data member value = 8

private member function value = 64

کی ورثہ This

کلاس کا ہر اوبجیکٹ اپنے ایڈریس کو بھی ایکسیس کر سکتا ہے۔ اس کے لئے پوائنٹر استعمال کیا جاتا ہے اور یہ this پوائنٹر اوبجیکٹ کا حصہ نہیں ہوتا۔ آپ یہ پوائنٹر اوبجیکٹ ڈیٹا ممبر اور مرکوز شکن دونوں کے لئے استعمال کر سکتے ہیں۔ اس پوائنٹر this کی تابع کیا ہو گی؟ یہ اوبجیکٹ کی تابع پر مخصوص ہے اس پوائنٹر کو آپ کیسے استعمال کر سکتے ہیں۔ آئیے دیکھتے ہیں۔

مثال نمبر 6.3

```
class exthis
{
public:
    exthis(int);
    void output();
private:
    int temp;
};

exthis::exthis(int t)
{
    temp = t*t;
}

void exthis::output()
{
    cout << "\n temp=" << temp;
    cout << "\n this->temp=" << this->temp;
    cout << "\n this, temp=" << (*this).temp;
}

void main()
{
    clrscr();
    exthis th(11);
    th.output();
}      //end main
```

ڈینا ممبر: static

کلاس کے ڈینا ممبر کی ہر کلاس اور جیکٹ کے لئے الگ الگ کالپنی ہوتی ہے یعنی کہ تمام اوزنکلش کے پاس ڈینا کی تفصیل الگ الگ ہوتی ہے۔ لیکن بعض اوقات تمام ویری اسٹبلو کی صرف ایک کالپنی تمام اوزنکلش کو فراہم کرنے کی ضرورت بھی پیش آ جاتی ہے۔ ایسی صورت حال کا سامنا کرنے کے لئے کلاس ویری اسٹبلز کو static ڈیکلائر کیا جاتا ہے۔ ایسے ویری اسٹبل صرف ایک واحد اینٹ شلائز کے لئے جاتے ہیں۔ یہ ویری اسٹبل کس طرح ڈیکلائر کئے جاتے ہیں۔ آئیے اس کی مثال دیکھتے ہیں۔

مثال نمبر 6.4 کلاس ڈینا ممبر static

```

class values
{
public:
values()
{
    ++temp;
}
static int temp; //declaring static variable
};

int values::temp = 2; //initialization

void main(void)
{
    values val1, val2;
    cout << "\n now value=" << val1.temp;
    {
        values val1, val2, val3
        cout << "\n now value of temp=" << val1.temp;
    }
    cout << "\n Again value=" << val1.temp;
    values val3;
    cout << "\n here value of temp=" << val3.temp;
    getch();
}

```

فونکشن ممبر: static

آپ نے مثال نمبر 6.1 میں کلاس ویری اسٹبل کو static ڈیکلائر کرنے کا طریقہ سیکھا ہے اس طرح آپ کلاس فونکشن کو بھی static ڈیکلائر کر سکتے ہیں۔

ہیں۔ آئیس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 6.5 کا سٹیشن

```
//static class functions
class example
{
public:
example( ) //Constructor
{
++temp;
}
~example( ) //Destructor
{
--temp;
}
static intfact( )
{
return temp;
}
private:
static int temp;
}
int example::temp=1;
void main(void)
{
clrscr();
cout << "value of temp=" <<example::fact();
example obj1, obj2;
cout << "\n temp=" <<example::fact();
}
cout << "\n temp outside scope=" <<example::fact();
example obj3, obj4;
cout << "\n temp=" <<example::fact();
```

آپریٹر اور لوڈنگ:

اس سے پہلے آپ نے فناشن اور لوڈنگ کے بارے میں پڑھا تھا۔ لیکن یہاں پر ہم فناشن کی بجائے آپریٹر کا ذکر کر رہے ہیں۔ آپ کو تقریباً 45 آپریٹر استعمال کرنے کی اجازت دیتا ہے۔ یہ آپریٹر C++ کی بنیادی نائپس کے لئے پہلے سے ڈیفائل کے گئے ہوتے ہیں لیکن جب آپ ایک کلاس کی بات کرتے ہیں یعنی ایک کلاس ڈیفائل کرتے ہیں تو آپ ایک نئی نائپ بنا رہے ہوتے ہیں۔ C++ کے بعض آپریٹر اس نئی نائپ کو استعمال کرتے ہوئے اور لوڈ کئے جاتے ہیں۔

درحقیقت آپریٹر اور لوڈنگ آپ کو C++ لینکوچ کو دوبارہ ڈیفائل کرنے کی ایک سہولت فراہم کرتا ہے۔ آپ سمجھتے ہیں کہ آپریٹر کی مدد سے آپ صرف جزو دکھنے کے لئے ہیں تو اس نہیں ہے بلکہ آپ ان کے کام کو اپنی ضرورت یا مرضی کے مطابق تبدیل کر سکتے ہیں۔

آپریٹر کی ورثہ:

آپ نے آپریٹر اور لوڈ کرنے کا فائدہ کر لیا ہے لیکن ایک سوال آپ کے ذہن میں ہو گا کہ ہم کون سا آپریٹر اور لوڈ کرنا چاہتے ہیں یعنی ہم ایک C++ آپریٹر کے یوں زیادتی آپریٹر بنانے کے ہیں۔ ان کے لئے C++ میں ایک کی ورثہ فراہم کرتی ہے جس کو استعمال کرتے ہوئے ہم مطلوب آپریٹر کی نشاندہی کر سکتے ہیں اور یہ کی وجہ آپریٹر (Operator) ہے۔ آئیے آپریٹر اور لوڈنگ کی ایک مثال دیکھتے ہیں۔ یہ بہت سادہ مثال ہے اس میں ہم یونی آپریٹر اور لوڈ کریں گے۔

مثال نمبر 6.6 آپریٹر کی ورثہ

```
class Example
{
private:
    int temp;
public:
    Example()
    {
        temp = 3;
    }
    int output()
    {
        return temp;
    }
    Example operator++()
    {
        ++temp;
    }
    Example over();
    over.temp=temp;
    return over;
}
```

```

void main(void)
{
    clrscr();
    Example obj1, obj2;
    cout << "\n obj1=" obj1.output();
    cout << "\n obj2=" obj2.output();
    ++obj1;
    obj2=++obj1;
    cout << "\n obj1=" obj1.output();
    cout << "\n obj2=" obj2.output();
    getch();
}

```

اور لوڈنگ بائنسی آپریٹر:

آپ نے اوپر مثال نمبر 6.6 میں یوں آپریٹر اور لوڈ کرنے کا طریقہ دیکھا اسی طرح آپ بائنسی آپریٹر بھی اور لوڈ کر سکتے ہیں۔ بائنسی آپریٹر کون کون سے ہیں اس کے بارے میں آپ تفصیل سے باب نمبر 1 اور 2 میں بڑھ کچے ہیں۔ آئیے سب سے پہلے آر جیمیک آسانٹ آپریٹر اور لوڈ کرنے کا طریقہ دیکھتے ہیں۔

مثال نمبر 6.7 اور لوڈنگ آر جیمیک آسانٹ آپریٹر

```

class exoverload
{
    int x,y;
public:
    exoverload()
    {
        x=0;
        y=0;
    }
    exoverload(int a, int b)
    {
        x=a;
        y=b;
    }
}

```

```

viiod input( )
{
    cout <<"\n Enter two integer values i.e 2,5:";
    cin >>x >>y;
}
//End of function

void output( )
{
    cout <<"\n x=" <<x <<"\n y=" <<y;
}
//End of output

void operator*=(exoverload);
};

//End of class

void exovererload::operator*=(exoverlod over)
{
    x*=over.x;
    y*=over.y;
}

void main(void)
{
    clrscr();
    exoverload over1;
    over1.input();
    cout <<"\n data output:\n";
    over1.output();
    exoverload over2(13, 5);
    cout <<"\n data with constructor passing parameter:";
    over2.output();
    over1*=over2;
    cout <<"\n after overloading:";
    over2.output();
    getch();
}

```

اس پروگرام میں ہم نے =* آپریٹر اور لوڈ کیا ہے۔ ایک کلاس ڈیفائن کی ہے جس کے دو کنسٹکٹر ہیں۔ ایک کنسٹکٹر میں کوئی پیرامیٹر پاس نہیں کیا

جبکہ دوسرا کنسٹرکٹر دو چیزیں پاس کرتا ہے۔ اس کے علاوہ ہم نے `(input)` فونکشن میں یوزر سے دو نمبر لیتوں ان پت لئے ہیں اور `operator*=(exoverload)` میں ہم نے اس آپریٹر کو اور لوڈ کیا ہے اور ایک اوبجیکٹ میں اس کا رزلٹ شور کروایا ہے۔

ملٹی پل اور لوڈنگ:

آپ نے اس سے پہلے یونیورسٹی اور بائسنسی آپریٹر اور لوڈ کرنے کا طریقہ پڑھا ہے۔ اب ہم آپ کو ایک ہی پروگرام میں دو یادوں سے زائد آپریٹر اور لوڈ کرنے کے بارے میں بتائیں گے۔ یہاں پر آپ آرٹھمیٹیک اور کمپریشن آپریٹر اور لوڈ کرنے کا طریقہ دیکھیں گے۔

مثال نمبر 6.8 کمپریشن آپریٹر اور لوڈنگ

```
class overloading
{
private:
    int x,y;
public:
    overloading()
    {
        x=4;
        y=7;
    }
    overloading(int a, int b)
    {
        x=a;
        y=b;
    }
    void input()
    {
        cout <<"Enter two integer values i.e 2,5:";
        cin >>x >>y;
    }
    void output()const
    {
        cout <<"\n x=" <<x <<"\n y=" <<y;
    }
    intput operator>(overloading);
};
```

```
{  
int res1=(x*y)/3;  
int res2=(over.x*over.y)/3;  
if(res1>res2)  
return 1;  
else  
return 0;  
}  
void main()  
{  
clrscr  
overloading over1;  
over1.input();  
overloading over2(13, 5);  
cout <<"\n Data output:\n";  
over1.output();  
//cout <<"\n Data with argument constructor";  
over2.output();  
if(over1>over2)  
cout <<"\n over 1 is greater than over2:";  
else  
cout <<"\n over 1 is less than over2:";  
//over1.output();  
getch();  
}
```

ہم نے اس پروگرام میں ایک فنکشن (function) int operator>(overloading) بنایا ہے۔ یہ ایک int ویلیووریٹر کرتا ہے جو کہ if(res1>res2) میں سے حاصل ہو رہی ہے۔

مشق

سوال نمبر 1: ان سوالات کے مختصر جواب تحریر کریں۔

(i) کس چیز کو زیرِ نظر کرتا ہے؟ *this

(ii) ان دونوں ڈیکھ لیشن میں کیا فرق ہے؟

xyz x(a);

xyz x=a;

(iii) آپ اس آپریشن *** کیس اور ٹوٹنگ کر سکتے؟

(iv) آرٹھمیک آپریٹر (+, -, *, /) فرینڈلی نیشن کے طور پر کیوں اور لوڈ کئے جاتے ہیں؟

(v) آپریٹر کی ورثہ کیسے استعمال کیا جاتا ہے۔

$$\frac{15}{9} + 1$$

سوال نمبر 2: آپریٹر کو اور لوڈ کرتے ہوئے اس مساوات کو حل کیں؟

سوال نمبر 3: پروگرام کے اس کوڈ میں کیا ایمپ ہے؟ فرض کریں کہ xyz ٹکسٹ میں لکھا ہے۔

```
xyz& xyz::operator=(const xyz&r)
```

```
{
```

```
x=r.x;
```

```
y=r.y;
```

```
return &this;
```

```
}
```

جوابات

1: جواب

- (i) کی ورڈ this اور جیکٹ کے لئے ایک پاؤنٹر ہوتا ہے اور جو اپنے ممبر فنکشن کو خود کا ال کرتا ہے یا یہ اور جیکٹ کو لیفٹ کرتا ہے اور یہ اس کے ممبر فنکشن کا ال کرتا ہے جس میں یہ اکپریشن لکھی ہو۔
(ii) اس میں پہلی دو ڈیلیٹ شن (a)x(y)z کا نتیجہ کر کو کا ال کرتی ہے جبکہ دوسری دو ڈیلیٹ شن xyz x=a; کا نتیجہ کر کو کا ال کرتی ہے۔
(iii) یہ علامت * آپریٹری طرح اور لوڈنگ میں ہو سکتی کیونکہ یہ C++ آپریٹر نہیں ہے۔
(iv) آر جیمیک آپریٹر زفرینڈ فنکشن کا طور پر اس لئے اور لوڈ کئے جاتے ہیں تاکہ ان کے باہمیں طرف والے آپریٹر کا نتیجہ دیکھیر کئے جاسکیں۔
(v) آپریٹر کی ورڈ فنکشن اس لئے اختیال ہوتا ہے کہ یہ اس فنکشن کی نشاندہی کرتا ہے جو کسی آپریٹر کو اور لوڈ کر رہا ہوتا ہے۔ مثلاً *operator "ab" کا مکر کو معلوم ہو گیا ہے کہ فنکشن *ab پہلے دو ڈیلیٹ شن کے لئے استعمال کیا گیا ہے۔

2: جواب

```

solving  $\frac{15}{9} - 1$ 

class question
{
    friend ostream& operator <<(ostream&, const question2);
public:
    question(int a=0, int c=1)
    {
        x=a;
        d=c;
    }
    question operator ++( );
private:
    int n,d;
};

question2 question2::operator ++( );
{
    n+=d;
    return *this;
}

```

```

}

ostream& operator <<(ostream&.str, const question2&q)
{
    return str <<q.n <<'/' <<q.d;
}

void main(void)
{
    clrscr();
    question2 q(15, 9), y=++q;
    cout <<"Answer of 15/q+1=" <<q;
    getch();
}

as mian ap ne ki or zuz this استعمال kiya hے jo ke object ke liye yeh nis hota as ko yeh lkhna hogा.
return *this;

```

3: جواب

باب نمبر 7

انہر پیپر اینڈ پولی مار فیزیم

اس باب میں ہم ادبیکٹ اور بینڈ پروگرامنگ کے دو اہم فچر انہر پیپر اینڈ پولی مار فیزیم کے بارے میں آپ کو بتائیں گے۔ انہر پیپر میں آپ ایک سافٹ ویر کو بار بار استعمال کرتے ہیں۔ اس میں نئی کلاسز پہلے سے بنی ہوئی کلاسز سے ڈرائیوڈ کی جاتی ہیں اور ان نئی کلاسز میں پہلی کلاس کی تمام وہ خصوصیات شامل ہوتی ہیں جن کو وہ اجازت دیتی ہے۔ ایک ہی سافٹ ویر کو بار بار استعمال کرنے سے آپ کا کوڈ آسان رہتا ہے اور اس میں ناممکن بچت ہوتی ہے۔ اس باب میں آپ مندرجہ ذیل عنوانات پر حصہ گے۔

کلاس Abstract

ورچوئل ڈسٹرکٹر

انہر پیپر کے لیواز

ملٹی پل انہر پیپر

فائل پر وسیلہ



کپوزیشن

انہر پیپر

انہر پیپر کی اقسام

اوور رائیڈنگ مبر نکش

کنسٹرکٹر اور ڈسٹرکٹر

ملٹی پل کلاسز

ورچوئل نکش



(Composition):

کپوزیشن:

یہ او بجیکٹ اوورینڈ (object oriented) پروگرامنگ کا ایک اہم فیچر ہے۔ ہمیں اکثر اپنی پہلے سے بنی ہوئی کلاسز کی مدد سے نئی کلاسز بنانی ہوتی ہیں۔ اس کے لئے C++ دو طریقے فراہم کرتا ہے۔ کپوزیشن اور انہریٹیشن۔ اس باب میں آپ یہ دونوں طریقے پڑھیں گے۔ لیکن پہلے، مم آپ کو کپوزیشن کے بارے میں بتاتے ہیں۔

کپوزیشن میں آپ ایک کلاس کی ذیلیشیں میں ایک یا ایک سے زائد کلاسز استعمال کرتے ہیں۔ جب آپ کی نئی کلاس کا کوئی ذیلی ممبر پہلی کلاس کا او بجیکٹ ہوتا ہے تو ہم کہتے ہیں کہ نئی کلاس کسی دوسرے او بجیکٹ کی کپوزٹ ہے۔ آپ یہ کام کس طرح کر سکتے ہیں۔ آئیے اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 7.1

```
#include<iostream.h>
#include<string.h>
#include<iomanip.h>
class Author
{
public:
    Author(char* a= " ", char* cou= " ")
        :name(a), country(cou){}
    void display name()
    {
        cout <<name;
    }
    void display country()
    {
        cout <<country;
    }
private:
    char* name, *country;
};
void main(void)
{
    clrscr();
    Author auth("M.Zulqurnain, "Pakistani");
    cout <<"The author of this book is";
    auth.displayname();
}
```

```

cout <<setw(2) <<"and he is a";
auth.displaycountry();
getch();
}

```

جب آپ اس پروگرام کو ایکسیٹ کریں گے تو یہ آٹھ پٹ نامہ سکرین پر ظاہر ہو گی۔

The author of this book is M.Zulquranin and he is a Pakistani.

(Inheritance):

انہریتینس:

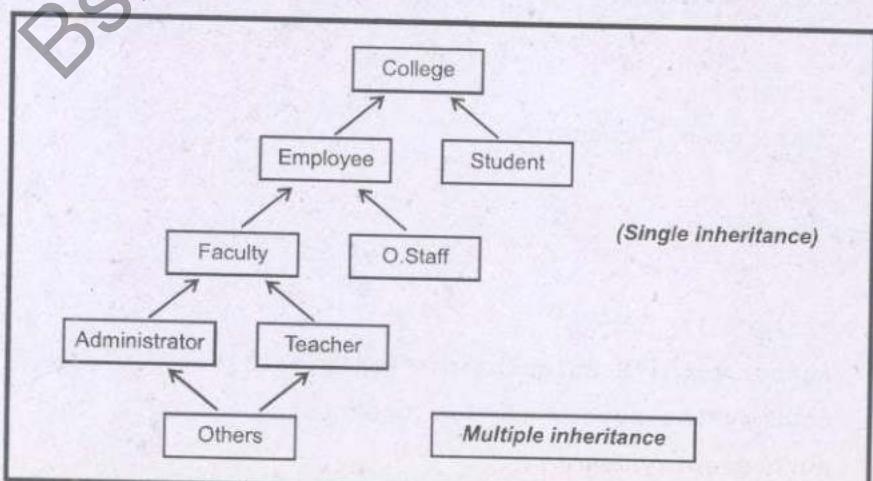
انہریتینس اور جیکٹ اور رینڈر کے مابین کا بہت سی بے بنی ہوئی کلاس سے نئی کلاس بنانے کے پر اس کو انہریتینس کہتے ہیں اور نئی کلاس ڈرائیورڈ (Derived) کلاس کہلاتی ہیں۔ ڈرائیورڈ کلاس میں اپنی بیس کلاس کی تمام خصوصیات ہوتی ہیں۔ اس کے علاوہ اس میں اپنی اضافی معلومات بھی ہو سکتی ہیں۔ یعنی انہریتینس کی مدد سے آپ پہلے سے بننے ہوئے سافت ویرز کو استعمال کرتے ہوئے نئے سافت ویرز بنانے سکتے ہیں۔ انہریتینس کی مدد سے آپ ایک کوڈ بار پار استعمال کر سکتے ہیں۔ مثلاً آپ نے ایک کلاس بنائی ہے اس کو ثیٹ کیا کہ کیا یہ درست کام کر رہی ہے یا نہیں؟ اب آپ کو جہاں بھی ایسا کام کرنا ہے تو اس کلاس کو انہریت کر دیں اور نئی کلاس میں اس کے ڈیماجبرز استعمال کریں۔ اس سے آپ کا نامممکن فتح جاتا ہے۔ اس کے علاوہ بجٹ بھی متاثر نہیں ہوتا اور اہم بات یہ کہ پروگرام سادہ رہتا ہے آپ دو قسم کی کلاس بنانے سکتے ہیں۔

(i) سُنگل کلاس انہریتینس (ii) ملٹی بیس کلاس انہریتینس

سُنگل انہریتینس میں ایک کلاس صرف ایک بیس کلاس سے ڈرائیورڈ ہوئی چہ جبکہ ملٹی بیس انہریتینس میں ایک کلاس کو کلاس کی ڈرائیورڈ کی جاتی ہے۔ ہم نے اوپر بھی بتایا تھا کہ ڈرائیورڈ کلاس میں اپنی بیس (base) کلاس کے علاوہ ایسے ڈیماجبرز اور میرنکشنز بھی ہو سکتے ہیں۔ یوں ایک ڈرائیورڈ کلاس اپنی بیس کلاس سے ہر ہی بھی ہو سکتی ہے۔ ایک ڈرائیورڈ کلاس میں اونچیکش بیس کلاس کی نسبت بہت جمع ہوتے ہوئے ہوتے ہیں۔

سُپر کلاس اور ڈرائیورڈ کلاس:

جب آپ ایک کلاس سے دوسری کلاس کو ڈرائیورڈ کریں گے تو ایک کلاس کا اوچیکٹ دوسری کلاس میں نہ لالہ ہوتا ہے اور اکثر ایک کلاس کا اوچیکٹ دوسری کلاس کا بھی اوچیکٹ ہوتا ہے۔ انہریتینس کس طرح کام کرتی ہے۔ اس کا ڈایاگرام ہم نے نیچے بنایا ہے۔



اوپر حرآرکی میں سٹوڈنٹس کلاس کا لج کا حصہ ہے لیکن آپ یہ کہہ سکتے ہیں کہ سٹوڈنٹس کلاس کا لج کلاس سے انہریٹ کی گئی ہے۔ اس کیس میں کلاس کا لج میں (base) کلاس ہے جبکہ سٹوڈنٹس ڈرائیور کلاس ہے۔ آپ نے انہرینیس کے بارے میں ابھی پڑھا ہے۔ آئیے اب اس کی ایک سادہ می مثال دیکھتے ہیں کہ C++ میں ہم ایک کلاس کی خصوصیات دوسری کلاس میں کیسے انہریٹ کر سکتے ہیں۔

مثال نمبر 7.2 کلاس انہرینیس پروگرام

```
#include<conio.h>
#include<iostream.h>
class base
{
private:
    int temp;
public:
    base(int);           //constructor
    void print(void);
};

class derived:public base
{
    //class inherited from base
private:
    int count;
public:
    derived(int, int);   //derived class constructor
    void show(void);

    void base::base(int a) //base class constructor
    {
        temp=a;
    }

    void derived::derived(int c, int d)
        //derived class constructor
        :base(d)           //call to base constructor
    {
        count=c;
    }
}
```

```

}

void base::print( )
{
    cout << "\n the initialized value=" << temp;
}

void derived::show( )
{
    cout << "\n Derived class:";

    cout << "\n The initialized value=" << count;
    print();
}

void main(void)
{
    clrscr();
    derived obj(12, 17);           // object of derived class
    obj.show();
    getch();
}

```

جب آپ اس پروگرام کو ایک سیکھو۔ کریں گے تو آپ کو یہ آئوٹ پڑ ملے گی۔

Derived class:

The initialized value = 12

The initialized value = 17

اس پروگرام میں ایک کلاس base ہے جس کا ڈائیاگرامبر temp پر ایجاد ڈیکلر کیا گیا ہے اور ممبر فنکشن میں کامن پرنٹ پلک ڈیکلر کے گئے ہیں۔ اس کے بعد ایک کلاس derived کے نام سے بنائی گئی ہے جو base کلاس سے انہریٹ کی گئی ہے۔ یعنی base کلاس کا کامن خصوصیات اس میں موجود ہیں۔ اس کلاس میں count ویری اسٹبل (ڈائیاگرامبر) پر ایجاد ہے جبکہ اس کلاس کا کنسٹرکٹر اور () show فنکشن پلک ڈیکلر کے ہیں۔ اس ڈرائیورڈ کلاس کے کنسٹرکٹر میں ہم نے میں کلاس کو اینی شلاز کرنے کے لئے اس کا کنسٹرکٹر کا ل کیا ہے۔

```

void derived::derived(int c, int d)
:base(d);

```

اس کے علاوہ پرنٹ فنکشن میں وہ ویری اسٹبل پرنٹ کیا ہے جو base کلاس کے کنسٹرکٹر میں اینی شلاز کیا گیا ہے جبکہ () show میں ڈرائیورڈ کلاس میں اینی شلاز کیا گیا ہے اور میں کلاس کا پرنٹ فنکشن بھی () show میں کال کیا گیا ہے۔

انہریٹینس کی اقسام:

اوپر آپ نے سادہ پروگرام لکھا جس میں ایک کلاس سے پہلے سے بنی ہوئی کلاس کی مدد سے بنائی گئی۔ یعنی یہ انہریٹینس کی ایک سادہ مثال ہے۔

تین قسم کی انہریٹینس پر فارم کرنے کی سہولت فراہم کرتا ہے۔

(i) Public (ii) Private (iii) Protected

آپ پیک اور پرائیویٹ کے بارے میں باب نمبر 5 تفصیل سے پڑھ چکے ہیں۔ اس باب میں آپ کا تعارف ایک نئے کی ورڈ protected سے کروائیں گے۔ پیک انہریٹینس میں ڈرائیوڈ کلاس کا ہر اوبجیکٹ میں کلاس کا بجیکت تصویر کیا جاتا ہے لیکن یہ درست نہیں ہے اور ایک ڈرائیوڈ کلاس اپنی میں کلاس کے پرائیویٹ ممبر ایکسیس نہیں کر سکتی۔ لیکن ایک ڈرائیوڈ کلاس اپنی سب کلاس کے protected ڈیٹا ممبر ایکسیس کر سکتی ہے۔ آجے اس کے لئے ایک پروگرام باتے یہ۔

مثال نمبر 7.3

```

class age
{
protected:
    int a;                                //protected data members
    char v[25];
public:
    age(int);
};

class name:public age                     //class inheritance
{
public:
    void show();
    name();
};

void age::age(int v)
{
    a=v;
}

void name::name()                         //base class constructor
:age(19)
{
    cout <<"\n Enter your name:";
```

```

    cin >> v;
}

void name::show( )
{
    cout << "\n Your age: " << a;
    cout << "\n Your name: " << v;
}

void main( )
{
    clrscr( );
    name n; //object of derived class
    n.show( );
    getch( );
}

```

اس پروگرام میں ہم نے کلاس age کے دو ڈیٹا ممبرز کو protected ڈیکلر کیا ہے اور ان کو پھر مزید ڈرائیوڈ کلاس name میں استعمال کیا ہے۔ اگر آپ ان ڈیٹا ممبرز کو پرائیویٹ ڈیکلر کرتے ہیں تو یہ ایردے گا۔ لفظاً کیا کلاس کے پرائیویٹ ڈیٹا ممبرز کو دوسرا کلاس ایکسیس نہیں کر سکتی۔ جب آپ اس پروگرام کو اجیز کیوٹ کریں گے تو یہ آؤٹ پٹ ڈپلے ہوگی۔

Enter your name : Cristina Maria

Your age : 19

Your name : Cristina Maria

(Overriding Member Functions):

آپ نے اس سے پہلے میں کلاس کے صرف ڈیٹا ممبرز (ویری ایبلز وغیرہ) ڈرائیوڈ کلاس میں استعمال کئے تھے اور آپ اس کے علاوہ میں کلاس کے ممبر فنکشن بھی ڈرائیوڈ کلاس میں استعمال کر سکتے ہیں۔ اور ان کا نام وہی ہوگا جو میں کلاس میں ہوگا۔ اس میں ایک بات بہت اہم ہے کہ آپ میں کلاس کا ممبر فنکشن بالکل اسی نام اور پیرامیٹرز کے ساتھ ڈرائیوڈ کلاس میں ڈیکلر کر رہے ہیں تو یہ فنکشن اور ایڈنڈ کہلانے گا لیکن آپ ڈرائیوڈ کلاس میں فنکشن کے پیرامیٹر تبدیل کر دیتے ہیں تو یہ فنکشن اور لوڈنگ ہوگی۔ اس کے بارے میں آپ باب نمبر 2 میں پڑھ چکے ہیں۔ اس لئے یہاں پر ہم صرف اور ایڈنڈ کا ذکر کریں گے۔

فرض کریں کہ ایک کلاس a آپ نے کلاس a سے ڈرائیوکی ہے اور a میں ایک فنکشن sum ہے اور یہی فنکشن آپ نے b کلاس میں بھی ڈیکلر کیا ہے تو یہ فنکشن اور ایڈنڈ ہے۔ اب آپ نے فنکشن sum کا ل کرنا ہے تو کس طرح آپ کون سا فنکشن کا ل کر سکتے ہیں اس کا ایک مخصوص طریقہ ہے۔

آگر آپ بعد میں لکھا گیا فنکشن کا ل کرنا چاہتے ہیں تو (b.sum) لکھیں گے اور پہلے والا فنکشن کا ل کرنے کے لئے (b.a::sum) لکھنا ہوگا۔ اسی طرح آپ اپنی ڈرائیوڈ اور میں کلاس میں ایک ہی نام کے ڈیٹا ممبرز بھی ڈیکلر کر سکتے ہیں اور ان کو ایکسیس کرنے کا طریقہ بھی اوپر کی طرح ہے اگر آپ ایک ڈیٹا ممبر temp کے نام سے دونوں کلاسز میں ڈیکلر کرتے ہیں تو b.temp شیٹ میٹ ڈرائیوڈ کلاس کو ایکسیس کرتی ہے جبکہ

آپ کی میں کلاس کے ڈیٹا ممبر کو ایکسیس کرنے کے لئے استعمال ہو گی اور اس کو ڈومینیٹ (dominate) کہتے ہیں۔ آئیے اس کے لئے ایک پروگرام لکھتے ہیں۔

مثال نمبر 7.4 اور رائیز گل کلاس میرنکشنز

```

class age
{
protected:
    int a;
public:
    void setage(int);
    void show();
};

class derived:public age
{
public:
    void show();
    //overriden function
};

void age::setage(int v)
{
    a=v;
}

void age::show()
{
    cout << "\n Base class member function:" ;
    cout << "\n Your age:" <<a;
}

void derived::show()
{
    cout << "\n Derived class member function:" ;
    cout << "\n Your age:" <<a;
}

void main(void)
{
    clrscr();
}

```

```

age a;
derived d;
a.setage(19);
d.setage(25);
a.show();
d.show();
d.age::show();
getch();
}

```

اس پروگرام میں ہم ایک کلاس age کے نام سے بنائی ہے اور ایک کلاس ڈرائیوڈ کے نام سے پہلی کلاس age سے ڈرائیوڈ کی گئی ہے اور ان دونوں کلاسز میں show() کا فنکشن لکھا ہے۔ یعنی کہ ڈرائیوڈ کلاس میں یہ فنکشن اور ڈرائیوڈ کیا گیا ہے اور بعد میں main() میں اسے کال کیا گیا ہے۔ main() میں دونوں کلاسز کے او بجیکٹ بھی لگتے ہیں۔ اور d.age::show() پر غور کریں اس سے کلاس کا ممبر فنکشن کال ہو رہا ہے لیکن ریفرنس ڈرائیوڈ کلاس کا ہے۔ اس پروگرام کی آڈیٹ پیش کرکے یوں ہو گی۔

Base class member function:

Your age : 19

Derived class member function:

Your age : 25

Base class member function:

Your age : 25

(Constructor and Destructor):

آپ اس بات سے بخوبی واقف ہیں کہ ایک ڈرائیوڈ کلاس اپنی بیس کلاس کے تمام ممبرز اسہر ہیث کرتی ہے اور جس ڈرائیوڈ کلاس کا او بجیکٹ بنایا جاتا ہے تو میں کلاس کا کنستراکٹر کال ہونا چاہئے جو میں کلاس کے ممبرز کو اپنی شلائیز کرے لیکن میں کلاس کے کنستراکٹر اور آسانی سے اس کو اس کا ڈرائیوڈ کیا کر سکتے ہیں۔ ہاں البتہ آپ کی ڈرائیوڈ کلاس کا کنستراکٹر اور آسانی سے آپ پریز میں میں کلاس کے آسانی سے آپ پریز اور کنستراکٹر کال کر سکتے ہیں۔ ڈرائیوڈ کلاس کا کنستراکٹر ہمیشہ میں کلاس کا کنستراکٹر کال کرتا ہے۔ اگر آپ ڈرائیوڈ کلاس کا کنستراکٹر نہیں لکھتے تو ڈرائیوڈ کلاس کا ڈیفائل کنستراکٹر میں کلاس کا ڈیفائل کنستراکٹر کال کرے گا لیکن ڈسٹرکٹر کال کرنے کا طریقہ اس کے الٹ ہے یعنی کہ ڈرائیوڈ کلاس کا ڈسٹرکٹر اس کی بیس کلاس کے ڈسٹرکٹر سے پہلے کال کیا جائے گا۔ اس کے لئے ہم نے ایک چھوٹا سا پروگرام خیچ کر کھا ہے اس پر غور کریں۔

مثال نمبر 7.5 کنستراکٹر اور ڈسٹرکٹر کا لئنگ

class base

{

```

public:
base( ):temp(4);                                //constructor
{
cout <<"\n Base class constructor:" <<temp;
~base( )                                         //destructor
{
cout <<"\n Base class desctructor:" <<temp;
}
protected:
int temp;
};
class derived:public base
{
derived( )                                     //constructor
{
temp++;
cout <<"\n Derived class destructor:" <<temp <<endl;
}
~derived( )                                     //destructor
{
temp=0;
cout <<"derived class destructor:" <<temp;
}
};

void main(void)
clrscr( );
derived d;
getch( );
}

```

اس پروگرام میں ڈرائیوڈ کلاس میں کلاس کے تمام ممبر ایمپھائز کر رہی ہے اور دونوں کلاسز کے ہم نے کنٹرول اور ڈسٹرکٹ کر لئے ہیں اور (main) میں ڈرائیوڈ کلاس کا اوبجیکٹ بنایا ہے۔ اس پروگرام کی آٹھ پٹ کچھ یوں ہو گی۔

Derived class constructor : 5

Derived class destructor : 0

Base class destructor : 0

اس آؤٹ پٹ پر غور کریں کہ میں کلاس کا ڈسٹرکٹر ڈرائیوڈ کلاس کے ڈسٹرکٹر کے بعد پرنٹ ہوا ہے یعنی ہر میں کلاس کا ڈسٹرکٹر اپنی ڈرائیوڈ کلاس سے پہلے اگر کبیوٹ ہوتا ہے۔ جبکہ ڈسٹرکٹر اس کے الٹ ہے۔ ڈسٹرکٹر ہمیشہ پہلے ڈرائیوڈ کلاس کا کال ہو گا۔

ملٹی پل کلاسز:

اس پیسے صرف ایک ڈرائیوڈ کلاس بناتے تھے جو میں کلاس کی تمام خصوصیات ایکسیس کر سکتی ہے۔ آپ ایک سے زیادہ کلاسز بھی بناتے ہیں۔ آئیے اس کے کوئی مثال دیکھتے ہیں۔

مثال نمبر 7.6 ملٹی پل کلاسز

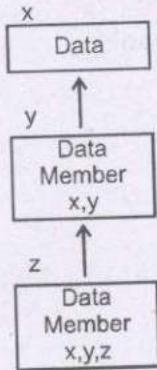
```
class x
{
    x( )
    cout <<"x class constructor:";
}
;

class y:public x
{
    y( )
    (cout <<\n y class constructor:");
}

class z:public x
{
    z( )
    (cout <<"\n z class constructor:");
};

void main(void)
{
    clrscr();
    z z1;
    getch();
}
```

اس طرح کے پروگرام کی حرآرکی کچھ یوں ہوتی ہے۔



یعنی ہماری دوسری کلاس **u** میں اپنے اور **x** کے دوسرے ممبرز شامل ہیں جن کی **x** اجازت دیتا ہے۔ اور **z** کلاس میں **x** اور **y** دونوں کی خصوصیات شامل ہیں۔

ورچوکل فنکشن:

ورچوکل کا مطلب ہے کہ ایک چیز کا ظاہر ہونا لیکن حقیقت میں وہ چیز نہ ہوتی۔ اسی طرح C++ میں بھی ورچوکل فنکشن کچھ اس طرح کام کرتا ہے جب آپ ورچوکل فنکشن بناتے ہیں اور ایک پروگرام میں اس کلاس کا ورچوکل فنکشن بناتے ہیں تو ہو سکتا ہے کہ حقیقت میں کسی مختلف (دوسری) کلاس کا فنکشن کال ہو رہا ہو۔ آپ ایک ورچوکل فنکشن کو میں کلاس میں ڈیکلائر کر سکتے ہیں۔ اور پھر سے کسی بھی ڈرائیوڈ کلاس میں ریئی یفاؤن کر لیں اور ایسے فنکشن کی ناپ اور آر گومنٹ ایک جیسے بھی ہو سکتے ہیں۔ اب یہ ڈرائیوڈ کلاس میں دوبارہ ڈیہن ان کا فنکشن میں کلاس میں لکھے ہوئے اس نام کے فنکشن کو اور ریئیز کر دے گا۔ ورچوکل فنکشن صرف مبرف فنکشن ہی ہو سکتے ہیں۔

آپ سوچ رہے ہوں گے کہ ورچوکل فنکشن کی ضرورت کیوں پیش آتی ہے۔ تو فرض کریں گے آپ کے یہ مختلف کلاسز کے کئی انجیکیشن ہیں لیکن آپ ایک ہی فنکشن کو کال کرتے ہوئے ان سب پر ایک ہی آپریشن پر فارم کرنا چاہتے ہیں تو ایسے کیس میں آپ کو کوئی فنکشن کی ضرورت پیش آتی ہے۔ ایک بات اور ذہن میں رکھیں کہ اگر آپ کے پاس دو فنکشن ایک ہی نام کے ہیں لیکن ان کے آر گومنٹ مختلف ہیں تو C++ انہیں مختلف فنکشن مانتی ہے اور وہ ورچوکل فنکشن کے طور پر کام نہیں کر سیں گے۔

یہ C++ کا بہت پاور فل فیچر ہے اور اس کو پولی مار فریم بھی کہتے ہیں۔ یہ ورچوکل فنکشن کی مدد سے پر فارم کی جاتی ہے اس کو آپ بعد میں پڑھیں گے۔ فی الوقت آئیے ورچوکل فنکشن کو استعمال کرتے ہوئے ایک پروگرام بناتے ہیں۔

مثال نمبر 7.7 سادہ ورچوکل فنکشن پروگرام

```

class base
{
public:
void display()
{
cout << "\n Base class virtual function:" ;
}

```

```

        }

};

class derive1 : public base
{
public:
    void display()
    {
        cout << "\n Derived1 calls, derive::display():";

    }
};

class derive2 : public base
{
public:
    void display()
    {
        cout << "\n derive2 class, derive2::display():";

    }
};

void main(void)
{
    clrscr(); base *b;
    derive1 d1;           //derive1 class object
    derive2 d2;           //derive2 class object
    base *p=&b;          //pointer to base class
    p->display();        //execute display
    p=&d1;                //put address of d1 in p
    p->display();        //execute display()
    p=&d2;
    p->display();
    getch();
}

```

ڈرائیڈ 1 اور ڈرائیڈ 2 کا بزر میں کلاس سے ڈرائیڈ کی گئی ہیں اور ان دونوں کا بزر میں ایک نکش (main) کا ہے۔ display() نکش میں ہم

visit our website: www.papersbook.Blogspot.com

نے ان دونوں کلاس کے اوچیکش d1 اور d2 بنائے ہیں اور میں کلاس کو پوازنٹ آسانی کیا ہے۔ اس کے بعد ہم نے اس پوازنٹ p کی مدد سے (display) کو کال کیا ہے۔ اس کے بعد ڈرائیوڈ d1 کا ایڈریس اس پوازنٹ p میں شور کر دیا ہے اور p کے ریفرنس (display) کو کال کیا ہے۔ d2 کے ساتھ بھی ایسا ہی کیا ہے۔ جب آپ اس پروگرام کو ایگرز میکس کریں گے تو یہ آٹ پٹ ہو گی۔

Base class virtual function

Base class virtual function

Base class virtual function

آپ نے دیکھا کہ تمام کال ایک ہی فنکشن کو جاری ہیں۔ اس نے ہیں کلاس کے فنکشن کے علاوہ کال کو نظر انداز کر دیا ہے کیونکہ میں کلاس کے لئے ہم نے پوازنٹ ڈیلکٹر کیا ہے۔ آپ تمام کوڈ یہی رہنے دیں صرف ایک لفظ کا اضافہ کر دیں اور پھر آٹ پٹ دیکھیں۔

```
class base
{
public:
    virtual void display()
    {
        cout << "\n Base class pure virtual function:";
```

اب دوبارہ اس پروگرام کو ایگرز میکس کریں اس کی آٹ پٹ تبدیل ہو چکی ہو گی۔ اب اس کی آٹ پٹ ہو گی۔

Base class virtual function

Derived class, Derivel::display();

Derived class, Derive2::display();

اس کو پولی مار فریم بھی کہتے ہیں۔ ایک فنکشن کو کال (display) کیا جا رہا ہے لیکن حقیقت مختلف کلاس کے مختلف فنکشن ایگرز میکس ہو رہے ہیں آپ نے پہلے جو پروگرام لکھا تھا وہ سادہ ورچوئل فنکشن کی ایک مثال تھی لیکن ایک لفظ ورچوئل لکھنے سے پروگرام کا تمام اصول تبدیل ہو گیا یہ ورچوئل فنکشن کی ایک مثال ہے۔ اس کے علاوہ آپ پہلے پروگرام میں یہ دو لائنز بے شک نہ لکھیں۔

base b;

base* p=&b;

لیکن دوسری دفعہ جب ورچوئل کی ورڈ لکھیں گے تو یہ لکھنا ہو گا۔

Abstract کلاس:

فرض کریں آپ نے ہیں کلاس میں ایک فنکشن ڈیلکٹر کیا ہے اور بعد میں آپ اس کلاس سے ڈرائیوڈ ہونے والی تمام کلاس میں اس فنکشن کو اور ایئڈ کرنا چاہتے ہیں تو پھر آپ اس ورچوئل فنکشن کو pure ورچوئل فنکشن بنالیں یعنی ہیں کلاس میں اس کی باڑی لکھنے کی ضرورت نہیں ہوتی یا دوسرے لفاظ میں اس کی باڑی ہیں کلاس میں نہیں لکھی جائے گی۔ یہ ایک pure ورچوئل فنکشن ایسا فنکشن ہوتا ہے جس کی اپنی کلاس (ہیں) میں کوئی باڑی نہیں

ہوتی۔ اس کا جزو طریقہ یہ ہے۔

virtual void x()=0;

تو یوں abstract کلاس ایسی کلاس ہوتی ہے جس میں کم از کم pure ورچوال ممبر فنکشن موجود ہو اور ایسی ڈرامیوز کلاس جس میں pure ورچوال فنکشن نہیں ہو گا وہ کنکریٹ (concrete) کلاس کہلاتی ہے۔

نوٹ: ہم نے آپ کو ایک بات پہلے بھی بتا دی ہیں کہ ہر ورچوال فنکشن ممبر فنکشن ہوتا ہے تو ہر کلاس کی ڈرامیوز کنکریٹ کلاس کو ورچوال ممبر فنکشن کی بادی (ڈیفائل) لازمی لکھنا ہوتی ہے۔ اس کا ہم نے ایک سادہ سارہ گرام پیچھے لکھا ہے تاکہ آپ کو بخوبی میں آسانی رہے۔

مثال نمبر 7.8 Abstract کلاس کی مثال

```
class base{
public:
    virtual void change(int)=0;           //virtual function
    virtual void display()=0;
protected:
    int i,c;
};

class derive1:public base
{
public:
    void change(int d)
    {
        i=d;
    }
    void display()
    {
        cout <<"\n Derive1.display( ) :" <<i;
    }
};

class derive2:public base
{
public:
    void change(int d)
    {
        i=d;
        i--;
    }
};
```

```

        }

void display( )
{
    cout << "\n derive2.display( ) : " <<i
}

};

void main(void)
{
    clrscr( );
    derive d1;
    derive d2;
    d1.change(7);
    d1.display( );
    d2.change(10);
    d2.display( );
    getch( );
}
}

```

اس پروگرام میں ایک Abstract کلاس بنائی گئی ہے جس کا نام base ہے اس کے دو ممبر فنکشنز ورچوئل ڈیکلیر کئے گئے ہیں۔ اس کے بعد دو مزید کلاسز بنائی ہیں جو اس base کلاس سے ڈرائیور ڈیگز کی گئی ہیں۔ ان میں ان فنکشنز کی ورثتی کو تحریر کی گئی ہے۔ جب آپ اس پروگرام کو الگز بیکٹ کریں گے تو اس کی آؤٹ پٹ یہ ہوگی۔

Derive1.display() : 7

Derive2.display() : 10

(Virtual Destructor):

آپ نے اس سے پہلے ورچوئل فنکشنز کے بارے میں پڑھا ہے کہ یہ ایے فنکشن ہوتے ہیں جنہیں میں کلاس میں ورچوئل ڈیکلیر کیا جاتا ہے اور سب کلاس میں یہ فنکشن اور اس کے جاتے ہیں۔ اس میں فنکشن کا نام اور جیسا میٹر ایک جیسے ہوتے ہیں۔ اس سے ظاہر ہوتا ہے کہ کنسٹرکٹر اور ڈسٹرکٹر ورچوئل ڈیکلیر نہیں کئے جاسکتے۔ یہ بات کنسٹرکٹر کے لئے درست ہے لیکن ڈسٹرکٹر کا میٹھا اس کو غلط قرار دیتا ہے۔ ہر کلاس کا ایک منفرد ڈسٹرکٹر ہوتا ہے اور آپ اسے ورچوئل بھی ڈیکلیر کر سکتے ہیں۔ آئیے اس کی ایک مثال دیکھتے ہیں۔

مثال نمبر 7. ورچوئل ڈسٹرکٹر

```

class super
{
public:
//~super( )

```

```

virtual ~super( );
{
cout << "\n Super destructor destroyed:" ;
}
};

class derived:public super
{
public:
derived( )
{
cout << "\n Derived destructor destroyed:" ;
}
};

void main(void)
{
cout << "\n Enter your information:" ;
p1.getname();
cout "\n //////////";
cout << "\n Employee information:" ;
emp.showname();
cout << "\n //////////";
cout << "\n Your data:" ;
p1.showname();
getch();
}

```

یہ ملی پل انہر پینس کی سادہ سی مثال ہے لیکن مثال خواہ سادہ ہو یا مشکل، ملی پل انہر پینس کا طریقہ ہیں جسے مطرح آپ ایک کلاس کو کئی کلاس سر سے بھی ڈرائیو کر سکتے ہیں اور یہ ڈرائیوڈ کلاس اپنی تمام ہیں کلاس سر کے مجرم انہر پینٹ کر سکتی ہے۔

انہر پینس کے لیوزر:

آپ بنے اور ابھی تک صرف سادہ انہر پینس پڑھی ہے۔ اس کے علاوہ بھی انہر پینس کے کئی طریقے ہیں مثلاً آپ ایسی کلاس بھی بناتے ہیں جو اسی کلاس سے ڈرائیوڈ کی گئی ہو جو مزید کسی کلاس کی سب کلاس ہے۔ یعنی آپ کی ہیں کلاس کسی اور کلاس سے ڈرائیوڈ کی گئی ہے۔ مثلاً

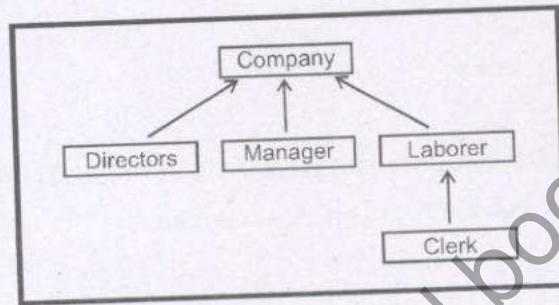
```

class x{ };
class y:public x{
//rest of code
class z:public y{

```

//code here

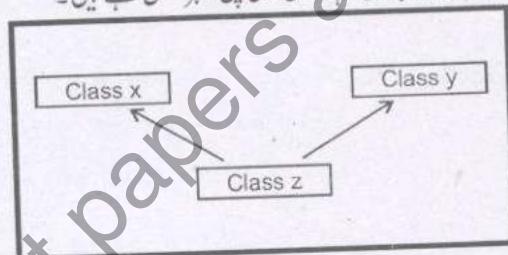
} اب یہاں پر y کلاس x سے ڈرائیور کی گئی ہے اور z کلاس y سے ڈرائیور کی گئی ہے۔ آپ اس کو حاصل کی مدد سے سمجھنے کی کوشش کریں کہ یہ کام کس طرح کام کرتی ہیں۔



ملٹی پل انہریٹنس:

(Multiple Inheritance):

ایک کلاس ایک سے زائد کلاس سے سمجھی ڈرائیور کی جائیجی میں کوئی پل انہریٹنس کرنے ہے۔



اس چارت سے ظاہر ہوتا ہے کہ کلاس z دو کلاس x,y سے ڈرائیور کی گئی ہے لہذا اس میں دونوں کلاسیں خصوصیات شامل ہیں۔ اس سے ظاہر ہوا کہ ملٹی پل انہریٹنس میں ایک ڈرائیور کلاس کی میں کلاس کے ممبرز کو انہریٹ کرتی ہے۔ یہ C++ کا بہت پاک نتیجہ ہے۔ آئیے ملٹی پل انہریٹنس کے لئے ایک پروگرام لکھتے ہیں۔

مثال نمبر 7.10 ملٹی پل انہریٹنس

```

#include<stdio.h>
const int size=45; //constant variable
class information
{
private:
char College[size];
char Degree[size];
public:
void input()
  
```

```

        gets(college);

        cout << "\n Enter Qualification:"
        << "\n (MBA, MCS, BSc, B.A):";
        cin >> degree;
    }

    void output( ) const
    {
        cout << "\n School or College name:" << college;
        cout << "\n Qualification:" << degree;
    }
};

class name
{
private:
    char name[size];
    int age;
public:
    void getname( )
    {
        cout << "\n Enter name:";
        gets(name);
        cout << "\n Enter age:";
        cin >> age;
    }
};

void showname( ) const
{
    cout << "\n Your name:" << name;
    cout << "\n Your age:" << age;
}

class employee:private name, private information
{

```

```
char hobb[size];
int temp;
void getname( )
{
    Name::getname( );
    cout <<"\n Enter your hobby:" ;
    gets(hobb);
    cout <<"\n Expected salary:" ;
    cin >>temp;
    information::input( );
}
void showname( ) const
{
    Name::showname( );
    cout <<"\n Your hobby:" <<hobb;
    cout <<"\n Your salary:" <<temp;
    information::output();
}
};

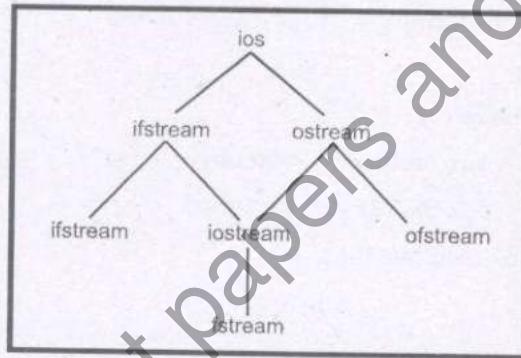
class personal:public Name
{
};

void main(void)
{
    clrscr( );
    employee emp;
    personal p1;
    cout <<"\n Enter Data for Employee1:" ;
    cmp.getname( );
    cout <<"\n //////////";
    getch( );
}
```

نے `fstream` کا او بجیکٹ `ofstream` بنایا ہے اور اس کے کنسٹرکٹر کو دو آر گومنٹس پاس کئے ہیں۔ ایک فائل کا نام اور دوسرا فائل موجود ہم نے فائل موجود و تحریر کئے ہیں۔ `ios` یعنی یہ فائل ان پت اور آؤٹ پت دونوں کے لئے ہیں۔ `ios::app` کا یہ فائل میں پہلے سے موجود ریکارڈ کو ذخیرہ نہیں کرتا اور متنے ریکارڈ فائلز کے آخر پر لکھتا ہے۔ اس کے علاوہ ہم نے یہ دیکھا ہے کہ فائل میں کل کتنے ریکارڈ ہیں اور اگر آپ کوئی ریکارڈ تلاش کرنا چاہتے ہیں تو وہ بھی کر سکتے ہیں۔

فائل پروسینگ:

`C++` میں فائل پروسینگ کے لئے آپ کو `<iostream.h>` اور `<fstream.h>` ہیڈر فائلز شامل کرنا ہوتی ہیں۔ `C++` میں تمام فائلز کو `ios` کی ایک تیزی سے جمع کی جاتی ہے اور ہر فائل کے اختتام پر اینڈ آف فائل کا مارک لگا ہوتا ہے یا پھر آخری ریکارڈ کا نمبر ہوتا ہے۔ جب ایک فائل اپن کی جاتی ہے تو اس کا او بجیکٹ `ios` اور ستریم جو اس کا او بجیکٹ سے منسلک ہوتی ہے وہ بھی اپن ہو جاتی ہے۔ `C++` فائلز پر کوئی سڑکھر عائد نہیں کرتی۔ اس کی وجہ سے اپن کیشن پر سڑکھر لائ گو کرنا ہوتا ہے۔ ان تمام کاموں کے لئے `C++` کی `io` کلاس استعمال کی جاتی ہے۔ اس کی حرارتی کچھ یوں ہے۔



I/O کی حرارتی

آپ کسی بھی فائل میں یہ `ios` کلاس کس طرح استعمال کر سکتے ہیں۔ آئیے اس پر گرام میں دیکھتے ہیں۔

مثال نمبر 7.11 ایک Sequential فائل بنانا

```

#include<iostream.h>
#include<fstream.h>
#include<stdlib.h>
#include<conio.h>
void main(void)
{
    clrscr();
    ofstream student("c://lib.dat, ios::out");
    if(!student)
        //overload!operator
  
```

```

        cerr("\n File could not be opened");

        exit(1);

    }

    cout << "\n Enter name, class, rollno:"
    << "\n For end press F6 or CTRL+Z:\n";

    char name[20], class1[7];

    int rno;

    while (cin >>name >>class1 >>rno)

    {
        student <<name' ' <<class1 <<' ' <<rno;
    }

    getch();
}

```

اب اس پروگرام کو ہم ایک نظر دیکھتے ہیں۔ اس میں ایک فائل اور پن کی گئی ہے اور فائل آؤٹ پٹ کے لئے کھوی گئی ہے۔ اس لئے ifstream اوبجکٹ کال کیا گیا ہے۔ اس اوبجکٹ کے لئے کٹر کو دو آر گو منس پاس کر رکھی ہیں۔ ایک فائل کا نام اور دوسرا فائل کا مود، کہ وہ کس مود میں اور پن کی گئی ہے۔ یہ مود کیا ہے آپ آگے پڑھیں گے۔ اگر یہ فائل اور پن ہو جاتی ہے تو ہمیں دوسرے کندیش ایگزکیوٹ ہو گی۔

اس کندیش میں ہم نے cerr استعمال کیا ہے یہ ایر متن آؤٹ پٹ کے لئے استعمال ہوتا ہے۔ اگر فائل اور پن نہیں ہو گی تو اس کندیش میں لکھا ہوا متن دلپٹے ہو گا اور پھر پروگرام ٹرمینٹ کر دیا جائے گا۔ ہم نے ان پٹ while لوب میں لی ہے اور یہ اس وقت تک ان پٹ لیتا رہے گا جب تک کہ آپ F6 کی بھی فائل کو اپن کرنے کے لئے آپ مندرجہ ذیل مود میں سے کسی بھی مود کا انتخاب کر سکتے ہیں۔ اس میں ساموڈ استعمال کرنا ہے۔ آئیے دیکھتے ہیں۔

وضاحت	مود
یہ تمام آؤٹ پٹ فائل کے آخر پر لکھتا ہے اس میں ڈیٹا فائل میں کہیں بھی لکھا جاسکتا ہے یہ فائل کو آؤٹ پٹ کے لئے اوپن کرتا ہے اور ڈیفالٹ کرس فائل کے آخر پر ہوتا ہے یہ آؤٹ پٹ کے لئے فائل اوپن کرتا ہے یہ فائل ان پٹ کے لئے اوپن کرتا ہے	ios::app ios::ate
اگر فائل پہلے نہیں ہو گی تو اوپن آپریشن ناکام ہو جائے گا یہ فائل میں تبدیلی نہیں کرنے دیتا یعنی اگر فائل پہلے سے موجود ہو گی تو یہ اوپن نہیں کرے گا	ios::out ios::in ios::nocreate ios::noreplace

آپ نے جو مثال پہلے دیکھی ہے وہ بہت سادہ مثال ہے۔ آئیے سبکو پہلے فائل بنانے کے لئے ایک اور پروگرام لکھتے ہیں۔

مثال نمبر 7.12 آٹھ پٹ اور ان پٹ فائل

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
#include<stdio.h>
class person
{
private:
    char name[20];
    int age;
public:
void showdata( )
{
    cout <<"\n Name:" <<name;
    cout <<"\n Age:" <<age;
}
void read( )
{
    cout <<"\n Enter Name:";
    gets(name);
    cout <<"\n Enter Age:";
    cin >>age;
}
};

void main( )
{
    clrscr();
    person p1;
    char ch;
    int no;
    fstream pfile;
    pfile.open(c:\\p1.dat", ios::app|ios::in);
    do
```

```

{
    cout << Enter Data: ";
    p1.read( );
    pfile.write((char*)&p1, sizeof(person));
    cout << "\n Do you want to continue: ";
    cin >> ch;
}

while(ch=='y'||ch=='Y');
pfile.seekg(0);
pfile.seekg(0, ios::end);
int space=pfile.tellg();
int trecord=space/sizeof(person);
cout << "\n There are" <<trecords <<"records in library";
cout << "\n Enter no to read: ";
cin >> no;
int position=(no-1)*sizeof(person);
pfile.seekg(position);
pfile.read((char*)&p1, sizeof(person));
p1.showdata();
{
    clrscr();
    super* ptr=new derived;
    delete ptr;
    getch();
}

```

اس پروگرام کو ایگزکوٹ کریں گے تو اس کی آؤٹ پٹ یہ ہوگی۔

Derived destructor destroyed :

Super destructor destroyed :

آپ نے دیکھا کہ دونوں کلاسز کے ڈسٹرکٹر ختم ہو گئے ہیں۔ اس میں ہم نے ایک لائن کو کو منش میں لکھا ہے۔

//~Super();

آپ اس کے کو منش ختم کر دیں اور اس سے نیچے والے درج ٹول ڈسٹرکٹر کو کو منش دیں اور پھر آؤٹ پٹ دیکھیں۔ ایسا کرنے سے صرف ایک ڈسٹرکٹر ختم ہو گا۔ اس کی آؤٹ پٹ یہ ہوگی۔

Super destructor destroyed :

مشق

سوال نمبر 1: ان سوالات کے جو اجواب دیں۔

- (i) کیا مارکسیم کیا ہے؟
- (ii) کیا Abstract کلاس کیا ہوتی ہے؟
- (iii) پوری اور انہرٹیٹیٹ میں کیا فرق ہوتا ہے؟
- (iv) ورچوئل اور پیور (Pure) ورچوئل میرنکشن کیا ہوتے ہیں؟

سوال نمبر 2: ایک پروگرام تحریر کریں جس میں دو کلاسز بنائی گئی ہوں اور تیسرا کلاس ان دونوں کلاسز سے ڈرائیڈ کی گئی ہو۔ اس پروگرام میں ایک کتاب سے متعلق تفصیل ہو جائی کہ اس کے میجرز اور قیمت وغیرہ کتنی ہے؟

جوابات

جواب : 1

- (i) پولی مارفیم میں آپ ایک سب کلاس اور اس کے میتھڈز کسی بھی کلاس کی حاصل کی جس کی میتھڈز کسی بھی کلاس کی حاصل کر سکتے ہیں اور یہ اس کلاس کے انٹرفیس کو ڈسٹریب نہیں کرتا لیکن اس کو تبدیل کرنے کی ضرورت نہیں ہوتی۔
- (ii) Abstract میں کلاس میں اس کلاس ہوتی ہے جس میں کم از کم ایک پیور (Pure) ورچوئل فنکشن شامل ہو اور اس کلاس کا آپ اوجیکٹ نہیں بن سکتے۔
- (iii) کلاسز کی کپوزیشن میں ایک کلاس دوسرا کلاس کا ممبر ڈیبلیر کیا جاتا ہے جبکہ انہر شیفس میں آپ ایک کلاس سے دوسرا کلاس ڈرائیور کرتے ہیں۔
- (iv) ورچوئل ممبر فنکشن ایسا فنکشن ہوتا ہے جو سب کلاس میں اور انہیں کیا جا سکتا ہے۔ جبکہ پیور ورچوئل فنکشن ڈائریکٹ کال نہیں کیا جا سکتا۔ ڈرائیور کلاس میں صرف اس کے اور انہیں کئے ہوئے فنکشن کا ل کے جا سکتے ہیں۔ اور یہ صفر (0) سے ایسی ٹھلاڑ کیا جاتا ہے۔

جواب : 2

```

const int month=5;

class JBD {
    private:
        char title[25];
        int cost;
    public:
        void input() {
            cout << "\n Enter title:" ;
            gets(title);
            cout << "\n Enter price:" ;
            cin >> cost;
        }
        void display() {
            cout << "\n Title: " << title;
            cout << "\n Price: " << cost;
        }
};

class shop {
    private:
        float sarr[month];
    public:
        void input();
        void display();
        cout << "\n Enter sale for 5 months:\n";
        for(int i=8; i<month; i++) {
            cout << "month ";
        }
}

```

```

    cin >>sarr[i];
}
}

void shop::display( )
{
    for(int i=8; i<months; i++)
        cout <<"\n Sales for month:" <<i+1 <<":")
        cout <<sarr[i];
}
}

class cbook:private JBD, private shop
{
private:
int page;
public:
void input( )
{
JBD::input( );
cout <<"\n Enter number of pages:";
cin >>page;
shop::input( );
}
void display( )
{
JBD::display();
cout <<"\n Total pages are:" <<page;
shop::display( );
}
}

void main
{
clrscr( );
cbook c1;
c1.input( );
c1.display( );
getch( );
}

```